



Apple Computer



To: Readers

From: Gail Pilkington

Re: Lisa Theory of Operations

Date: June 3, 1983

-----

This memo covers the latest revision of the manuscript for the manuals titled *Lisa Theory of Operations* (LTOO) and *Lisa Hardware Manual* (LHM). The LHM includes:

- o Chapters 1, 2, and 3 only,
- o Condensed Preface
- o No appendixes.

The LTOO manual will include everything in the version before you, schematics in the appendixes, and a complete index.

### Background

The manuscript was written by a consultant and reviewed by Paul Baker and others. I edited the files to correct the errors and omissions, eliminate ambiguous and inconsistent terminology, and generally clean things up. I used an indented-outline format, which will not be used for the final books, and I added space in the manuscript for line-art, wrote a table of contents, and compiled this version. Time was not spent preparing an index for this draft because it would not be correct for the final books, which will be printed very soon.

The art in this version is the original art, which I hand-edited; line-art will be produced by a consulting firm: CBM. In this manuscript, line-art is not embedded. Instead, a page that provides space for line-art and the figure caption is labeled with an "a" and the original art appears on a separate page labeled "b"; see the table of contents.

### Notations in this manuscript

Because the manuscript is written in Pascal, some special notations appear in the text as an instruction to either the typesetter or, should we decide not to typeset the books, to the person who uses LisaWrite to produce camera-ready copy. Following are definitions and examples of the notations, and examples of how the notations will look in the finished books.

1. Over bar: A slash mark followed by a space represents a bar over preceding characters. For example, VMA/ becomes  $\overline{VMA}$ , B/L/ becomes  $\overline{B/L}$ , etc.
2. Binary, Hex, and Base 10: @2, @16, and @10 represent a subscripted 2, 10, or 16. For example, FF@16 becomes  $FF_{16}$ , 00E010@16 becomes  $00E010_{16}$ , etc.
3. Multiplication sign: An exclamation point followed by a period, (!.), represents a raised multiplication sign. For example, SLOT/!.A20/!.A19 becomes  $SLOT/\cdot A20/\cdot A19$



THE SECRETARY OF THE ARMY  
WASHINGTON, D. C.

TO THE SECRETARY OF THE ARMY  
FROM THE SECRETARY OF THE ARMY

THE SECRETARY OF THE ARMY  
WASHINGTON, D. C.

TO THE SECRETARY OF THE ARMY  
FROM THE SECRETARY OF THE ARMY

THE SECRETARY OF THE ARMY  
WASHINGTON, D. C.

TO THE SECRETARY OF THE ARMY  
FROM THE SECRETARY OF THE ARMY

THE SECRETARY OF THE ARMY  
WASHINGTON, D. C.

TO THE SECRETARY OF THE ARMY  
FROM THE SECRETARY OF THE ARMY

THE SECRETARY OF THE ARMY  
WASHINGTON, D. C.

TO THE SECRETARY OF THE ARMY  
FROM THE SECRETARY OF THE ARMY

THE SECRETARY OF THE ARMY  
WASHINGTON, D. C.



## Lisa Theory of Operations

WARNING: This equipment has been certified to comply with the limits for a Class B computing device, pursuant to Subpart J of Part 15 of FCC rules. Only peripherals (computer input/output devices, terminals, printers, and the like) certified to comply with the Class B limits may be attached to this computer. Operation with noncertified peripherals is likely to result in interference to radio and TV reception.

{copyright notice}



17.

18.

19.

20.

21.

22.

23.

24.

25.

26.

27.

28.

29.

30.

31.

32.

33.

34.

35.

36.

37.

38.

39.

40.

41.

42.

43.

44.

45.

46.

47.

48.

49.

50.

51.

52.

53.

54.

55.

56.

57.

58.

blank

59.

60.

61.

62.

63.

64.

65.

66.

67.

68.



## CONTENTS

---

Preface	xiii
Chapter 1. Architecture	1- 1
1.1 Layout	1- 1
1.2 Hardware Structure	1- 3a
1.3 The Central Processing Unit	1- 4
1.4 Memory	1- 5a
1.5 Internal Buses	1- 7
1.6 Data Storage Media	1- 8
1.7 User Interfaces	1- 8
1.8 Additional Features	1- 8
Chapter 2. Programming	2- 1
2.1 The Instruction Set	2- 1
2.2 CPU Registers and Their Use	2- 2
2.3 Memory Management Scheme	2- 2
2.3.1 Address Transformation	2- 4
2.3.2 The MMU Registers	2- 5a
2.3.3 MMU Initialization	2- 8
2.3.4 Contexts	2- 9
2.4 Addressing i Special I/O Space	2-10
2.5 I/O Map	2-10
2.5.1 Floppy Disk Control	2-11
2.5.2 Serial Port Control	2-15
2.5.3 Parallel Port Control	2-16
2.5.4 Keyboard/Mouse Control	2-18
2.5.5 Processor Board Control	2-25
2.6 Interrupt Handling	2-26
2.7 Error Processing	2-28
2.8 Status Register	2-29
Chapter 3. Internal Specifications	3- 1
3.1 Packaging	3- 1
3.2 Specifications	3- 3a
3.2.1 Environmental	3- 4
3.2.2 Physical	3 -4

## CONTENTS, continued

---

3.2.3	Electrical	3- 4
3.3	Modules	3- 5
3.1.1	Cabinet	3- 5
3.3.2	Power Supply	3- 6a
3.3.3	Floppy-Disk Drives	3- 6a
3.3.4	Printed Circuit Boards	3- 7
3.3.5	Keyboard and Mouse Assemblies	3- 7
3.4	Expansion Bus	3- 7
3.4.1	Bus Signal List	3- 8
3.4.2	Bus Signal Descriptions	3- 9
3.4.3	Bus Parameters	3-12
3.4.4	Bus Timing	3-13
3.4.5	DMA Scheme	3-15a
3.4.6	Bootstrap Protocol	3-16
3.5	The External Ports	3-19
3.5.1	Serial Port Interface	3-20
3.5.2	Parallel Interface Port	3-20
3.5.3	The Mouse Interface	3-22
3.5.4	The Keyboard Interface	3-23
3.5.5	Composite Video Interface	3-23
Chapter 4. The Processor Board		4- 1
4.1	Processor Board Block Diagram	4- 1
4.1.1	CPU Access to Memory	4- 3
4.1.2	Video Access to Memory	4- 4a
4.1.3	CPU Access to I/O	4- 5a
4.1.4	I/O Access to DMA Address Latch	4- 6a
4.1.5	I/O Access to Memory (DMA)	4- 7a
4.2	Instruction Cycle and Timing	4- 9a
4.2.1	Internal Timing	4- 9a
4.2.2	Memory Management Timing	4-10
4.2.3	Memory Timing	4-12
4.2.4	Video Timing	4-14a
4.3	The Central Processing Unit (CPU)	4-15a
4.3.1	Clock Generation	4-16a
4.3.2	Processor Control Signals	4-17
4.3.3	Address and Data Lines	4-17
4.3.4	Bootstrap ROM	4-18
4.4	The Memory Magement Unit (MMU)	4-18
4.4.1	MMU RAM Storage	4-20



## CONTENTS, continued

---

4.4.2	SOR and SLR Initialization	4-23
4.4.3	Address Translation	4-24
4.4.4	Memory Timing Generation	4-25
4.5	Video Control	4-26
4.5.1	Video Address Counter	4-27 <sup>a</sup>
4.5.2	Video Data Shift Register	4-27 <sup>a</sup>
4.5.3	Video State Machine	4-28
4.5.4	Video Page Register	4-29
4.6	Bus Interfaces	4-29
4.6.1	System Bus Interface	4-30 <sup>a</sup>
4.6.2	Memory Bus Interface	4-33
4.6.3	The Video Interface	4-34
4.7	Decode and Latches	4-35
4.7.1	I/O Decode	4-35
4.7.2	Processor Board Control Register	4-36
4.7.3	Memory Error Address Latch	4-37
4.7.4	Processor Board Status Latch	4-37
4.7.5	Time Delay Logic	4-38
Chapter 5. Memory Boards		5- 1
5.1	Memory Block Diagram	5- 2 <sup>a</sup>
5.2	Row and Column Addressing	5- 3
5.2.1	Address Lines	5- 3
5.2.2	Slot Decode	5- 3
5.2.3	Matrix Device Decode	5- 4
5.2.4	Matrix Address Strokes	5- 5
5.3	Data and Parity	5- 6
5.3.1	Memory Data Lines	5- 6
5.3.2	Memory Parity	5- 7
5.3.3	Memory Refresh	5- 7
5.4	Memory Timing	5- 9 <sup>a</sup>
5.4.1	Row Selection Timing	5- 9 <sup>a</sup>
5.4.2	Address Multiplex Timing	5-10 <sup>a</sup>
5.4.3	Data and Parity Timing	5-11 <sup>a</sup>
Chapter 6. The I/O Board		6- 1
6.1	I/O Board Block Diagram	6- 1
6.2	Floppy-Disk Controller	6- 2 <sup>a</sup>

## CONTENTS, continued

---

6.2.1	Processor and Memory	6- 4
6.2.2	System Bus Interface	6- 6
6.2.3	Timing Generation	6- 6
6.2.4	Disk State Machine	6- 9a
6.2.5	Stepper Motor Control	6-11
6.2.6	General Drive Control	6-11
6.3	Floppy-Disk Controller Operation	6-12
6.3.1	Disk Processor Operation	6-32a
6.3.2	Disk Macro Commands	6-34
6.3.3	Data Encoding and Decoding	6-37
6.3.4	Disk Formatting	6-38a
6.3.5	Disk State Machine Operation	6-41
6.4	The Serial I/O Controller	6-45
6.4.1	The Serial-Port Controller	6-45
6.4.2	The Serial Ports	6-46
6.4.3	Baud Rate Generation	6-46
6.4.4	Serial Port Operation	6-46
6.5	Parallel-Port Controller	6-46
6.5.1	68000 Bus Interface for the Parallel-Port Controller	6-47
6.5.2	Parallel Port Interface	6-47
6.5.3	Parallel Port Operation	6-49
6.5.4	Parallel Port Timing	6-49
6.6	The Keyboard/Mouse Controller	6-49
6.6.1	68000 Bus Interface for the Keyboard/Mouse Controller	6-49
6.6.2	The COPS Processor	6-50
6.6.3	Keyboard/Mouse Interface	6-51
6.6.4	Software On-Off and Reset Logic	6-52a
6.6.5	Other Control Lines	6-52a
6.7	Miscellaneous Logic	6-53
6.7.1	Speaker Volume Control	6-53
6.7.2	Battery Supply and Control	6-54
6.7.3	Video Contrast Latch	6-55
 Chapter 7. The Video Board		 7- 1
7.1	Block Diagram	7- 1
7.2	Power Supply Circuits	7- 2a
7.2.1	The +12 VDC and +5 VDC Supplies	7- 3

## CONTENTS, continued

---

7.2.2	The +33 VDC Supply	7- 3
7.3	The Video-Amplifier Circuit	7- 3
7.3.1	Video-Data Input and Contrast Control	7- 4
7.3.2	Cathode Drive Circuit	7- 4
7.4	Vertical Deflection Circuit	7- 5
7.4.1	Vertical Deflection Oscillator	7- 5
7.4.2	Vertical Voltage Amplifier	7- 5
7.4.3	Vertical Power Amplifier	7- 6
7.4.4	Bootstrap Circuit	7- 7
7.5	Horizontal Deflection Circuits	7- 7
7.5.1	Horizontal Input Circuit	7- 7
7.5.2	Horizontal Sweep Amplifier	7- 8
7.5.3	Horizontal Deflection	7- 8
7.5.4	Horizontal Flyback	7- 9
7.5.5	Overvoltage Crowbar	7- 9
7.5.6	Brightness and Focus	7- 9
Chapter 8. User Interfaces		8- 1
8.1	The CRT	8- 1
8.2	The Keyboard	8- 1
8.2.1	Keyboard Logic	8- 1
8.2.2	Keyboard Timing	8- 3
8.2.3	Keyboard Interface	8- 3
8.3	The Mouse	8- 4a
8.3.1	Internal Components	8- 4a
8.3.2	The Mouse Interface	8- 5a
8.4	Other User Controls	8- 6
8.4.1	The On-Off Switch	8- 6
8.4.2	The RESET Switch	8- 6
8.4.3	The Speaker	8- 6
8.4.4	The Composite Video Output	8- 7
Chapter 9. Floppy-Disk Drives		9- 1
9.1	Specifications	9- 1
9.1.1	Media	9- 1
9.1.2	Speed	9- 1
9.1.3	Electrical	9- 1



## CONTENTS, continued

---

9.1.4	Environmental	9- 2a
9.2	Drive Block Diagram	9- 2a
9.3	Drive Interface	9- 2a
9.3.1	Drive-Interface Signals	9- 3
9.3.2	Drive-Interface Timing	9- 3
9.4	Basic Drive Operation	9- 5
9.4.1	Inserting and Removing Disks	9- 5
9.4.2	Loading Disk Heads	9- 6
9.4.3	Positioning Heads	9- 6
9.4.4	Data Read/Write	9- 7
Chapter 10. User Interfaces		10- 1
10.1	Power Supply Block Diagram	10- 2
10.2	AC Input Circuits	10- 3a
10.2.1	AC Line Connection	10- 4
10.2.2	On-Off Control	10- 4
10.2.3	Primary Rectification	10- 4
10.3	The Flyback Oscillator	10- 5a
10.3.1	Flyback Starting Bias	10- 6
10.3.2	Flyback Oscillator Operation	10- 6
10.3.3	Flyback Control Circuit	10- 7a
10.4	DC Output Circuitry	10- 9
10.4.1	DC Voltage Outputs	10- 9
10.4.2	DC Voltage Controls	10-10
10.5	Standby and Auxiliary Video Circuits	10-11
10.5.1	The Standby Supply	10-11
10.5.2	The Video Controls	10-11
Chapter 11. Assemblies		11- 1
11.1	User-Replaceable Components	11- 1
11.2	CRT Assembly	11- 2
11.2.1	CRT Access	11- 2
11.2.2	Deflection Yoke	11- 3
11.2.3	Video Board	11- 3
11.2.3	Flyback Transformer	11- 4
11.3	Cabling	11- 5
11.3.1	Power Cabling	11- 5

## CONTENTS, continued

---

11.3.2 Disk Cabling	11- 6
11.3.3 Interface Connections	11- 7
11.3.4 Logic Connections	11- 7
11.4 Disk Drive Assembly	11- 7
11.5 Miscellaneous Assemblies	11- 7
11.5.1 Speaker Assembly	11- 8
11.5.2 On-Off Switch Assembly	11- 8
11.5.3 Motherboard Assembly	11- 8
11.6 Keyboard and Mouse	11- 9
11.6.1 Keyboard Assembly	11- 9
11.6.2 Mouse Assembly	11-10

## Appendixes

---

Appendix A. Processor Board Schematic	A- 1
Appendix B. Memory Board Schematic	B- 1
Appendix C. I/O Board Schematic	C- 1
Appendix D. Video Board Schematic	D- 1
Appendix E. Keyboard Schematic	E- 1
Appendix F. Power Supply Schematic	F- 1
Appendix G. Motherboard Schematics	G- 1
Appendix H. Keyboard Layouts	H- 1
Appendix I. Video Assembly Schematic	I- 1
Appendix J. Radio and Television Interference	J- 1

Glossary	G- 1
----------	------

---

Index <i>coming soon</i>	I- 1
--------------------------	------

---

## List of Figures

---

1-1 Front View	1- 2a
1-2 Back View with Panel Removed	1- 2b
1-3 Block Diagram	1- 3b
1-4 Processor Block Diagram	1- 5b
1-5 Physical Memory Map	1- 7
2-1 Address Word Decode	2- 3b

## CONTENTS, continued

---

2-2	Segment Limit Check	2- 5b
2-3	Special I/O Addressing	2- 6
2-4	MMU Access Control Bits	2- 7
2-5	I/O Space Overview	2-11
2-6	Floppy-Disk Commands	2-12
2-7	Floppy-Disk Command Block	2-13
2-8	Floppy-Disk Controller Error Codes	2-14
2-9	Floppy-Disk Interrupt Source	2-15
2-10	Serial Port Baud Rates	2-16
2-11	Parallel Port Bit Correspondance	2-17
2-12	Parallel Port Addressing	2-18
2-13	Keyboard and Mouse Addressing	2-19
2-14	Keyboard COPS Commands	2-20
2-15	Keyboard Codes	2-21b
2-16	Keyboard COPS Reset Codes	2-23
2-17	Processor Board Control Addressing	2-26
2-18	Exception Vector Table	2-27
2-19	Status Register	2-30
3-1	Modules in Cabinet	3- 3b
3-2	Cabinet Assembly	3- 6b
3-3	Signal Pin Layout	3- 9
3-4	Expansion Bus Read-Timing Diagram	3-14 b
3-5	Expansion Bus Write-Timing Diagram	3-15b
3-6	Bootstrap ROM Format	3-17
3-7	Serial Port Pin Assignments	3-20
3-8	Parallel Port Pin Assignments	3-21
3-9	Mouse Interface Pin Assignment	3-23
4-1	Processor-Board Block Diagram	4- 2b
4-2	CPU Access To Memory	4- 4b
4-3	Video Access To Memory	4- 5b
4-4	CPU Access To I/O	4- 6b
4-5	I/O Access To DMA Address Latch	4- 7b
4-6	I/O Access To Memory (DMA)	4- 8b
4-7	Processor Board Internal Timing	4- 9b
4-8	MMU Timing Diagram	4-11b
4-9	Memory Control Timing Diagram	4-13b
4-10	Video Control Timing Diagram	4-14b
4-11	Video Page Timing	4-15b
4-12	Processor Board Timing Generation	4-16b
4-13	MMU Block Diagram	4-19b



## CONTENTS, continued

---

4-14	MMU Memory Configuration	4-21b
4-15	Video Control Block Diagram	4-27b
4-16	Processor Board Bus Interfaces	4-30b
5-1	Memory Block Diagram	5- 2b
5-2	Memory Address Decoding	5- 4
5-3	Memory Refresh Pattern	5- 8
5-4	Memory Timing Diagram	5- 9b
5-5	Memory Row Address Timing	5-10b
5-6	Memory Data and Parity Timing	5-11b
6-1	I/O Board Block Diagram	6- 2b
6-2	Floppy-Disk Controller Block Diagram	6- 3b
6-3	Floppy-Disk Controller Address Space	6- 5b
6-4	Floppy-Disk Counter Timing States	6- 7b
6-5	Disk State Machine	6- 9b
6-6	Floppy-Disk Controller Commands	
	a. Handshake	6-13
	b. Read	6-14
	c. Write	6-15
	d. Unclamp	6-16
	e. Format	6-17
	f. Verify	6-18
	g. Format Track	6-19
	h. Verify Track	6-20
	i. Read Brute Force	6-21
	j. Write Brute Force	6-22
	k. Clamp	6-23
	l. Seek	6-24
	m. User Program	6-25
	n. Clear Interrupt Status	6-26
	o. Drive Enable	6-27
	p. Drive Disable	6-28
	q. ROM Wait	6-29
	r. Go Away	6-30
6-7	Floppy-Disk Macro Command Flowchart	6-31b
6-8	Floppy-Disk Controller I/O Block	6-32b
6-9	Floppy-Disk Controller Error Codes	6-37
6-10	Floppy-Disk Data Encoding Scheme	6-38b
6-11	Lisa Disk Format	6-39b

## CONTENTS, continued

---

6-12	Seek Flowchart	6-43b
6-13	Keyboard Data Format and Timing	6-52b
7-1	Video Board Block Diagram	7- 2b
8-1	North American Keyboard Layout	8- 2b
8-2	Keyboard Interface	8- 4b
8-3	Mouse Movement Waveforms	8- 5b
9-1	Drive Block Diagram	9- 2b
9-2	Drive-Interface Signals	9- 3
9-3	Carriage Movement Timing	9- 4b
10-1	Supply Current Output	10- 1
10-2	Power Supply Block Diagram	10- 3b
10-3	Flyback Oscillator Circuit	10- 5b
10-4	Flyback Oscillator Waveforms	10- 7b
10-5	Flyback Control Block Diagram	10- 8b

## CHAPTER 4. THE PROCESSOR BOARD

---

The heart of the Lisa is the processor board. It contains the following major logic components:

- \* Central processing unit (CPU)
- \* Memory management unit (MMU)
- \* Timing generation
- \* Memory timing
- \* Video generation
- \* Interrupt control
- \* I/O decode
- \* Direct memory access (DMA), error address, and flag latches.

The processor board is so called because it contains the logic associated with the 68000 CPU. It also contains the main memory management and bus control functions for the Lisa. Some additional circuitry, such as error latches and video control, are also located on this board for convenience.

The function of the processor board is to execute the Lisa software, provide main timing for communication within the Lisa, and to provide additional control functions for the video logic.

### 4.1 Processor Board Block Diagram

An overview of how the components on the processor board logically interact is shown in Figure 4-1. Refer also to Figure 1-2 in Chapter 1, which gives an overview of the Lisa in block diagram form.





Figure 4-1. Processor Board Block Diagram

The processor board is capable of performing a number of operations that involve data flow within the board. Since both processor and video memory cycles are interleaved within one machine cycle, this leads to two major data paths during operation. Additionally, data transfer to and from peripheral devices during normal or DMA cycles gives rise to two other cycles, plus the path that loads the DMA address latch.

Addresses generated by the CPU are translated from logical to physical addresses by means of the MMU. A discussion of MMU programming can be found in section 2.3. Addresses generated by the video or the DMA logic are not subject to translation, i.e., they are physical addresses.

ROM provides non-volatile storage that is used during power-up initialization for bootstrap of the operating system. It can be addressed either with or without use of the MMU.

Internal timing on the board and on the bus conforms to the timing requirements of the 68000, as described in the 68000 User's Manual. Operation of RAM memory is controlled by additional timing logic on the processor board.

The data-flow paths within the processor board are

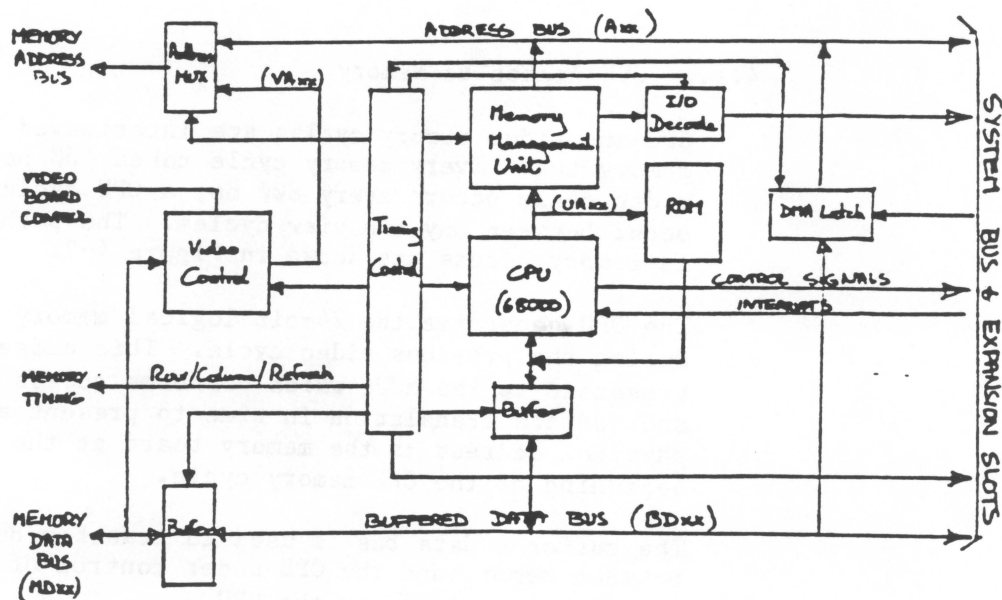


Figure 4-1. Processor Board Block Diagram

P.P.

classified as follows:

- \* CPU memory access
- \* Video memory access
- \* CPU access to I/O
- \* I/O access to the DMA address latch
- \* I/O access to memory.

Video memory access and I/O memory access can occur simultaneously because of the separation of the control lines and data paths involved. These classes are discussed in the following subsections.

#### 4.1.1 CPU Access to Memory

CPU and video memory cycles are interleaved on the memory bus. Every memory cycle takes 400 ns. A video cycle occurs every 800 ns; a CPU cycle can occur between any two view cycles. The paths used in memory access are shown in Figure 4-2.

The CPU generates the 24-bit logical memory address during the previous video cycle. This address is presented to the MMU, which performs access checks and address translation in time to present a physical address to the memory board at the beginning of the CPU memory cycle.

The buffered data bus is used to transfer data between memory and the CPU under control of bus signals manipulated by the CPU.



Figure 4-2. CPU Access to Memory

#### 4.1.2 Video Access to Memory

The video cycle is used to access the bit-mapped display data located in main memory. The paths used by the video control to access memory data are shown in Figure 4-3.

The video control generates physical addresses that are presented to the memory addressing multiplexer on the processor board for direct access to the memory location. Data that is read out of memory in this way is presented to the video control via the memory data bus.



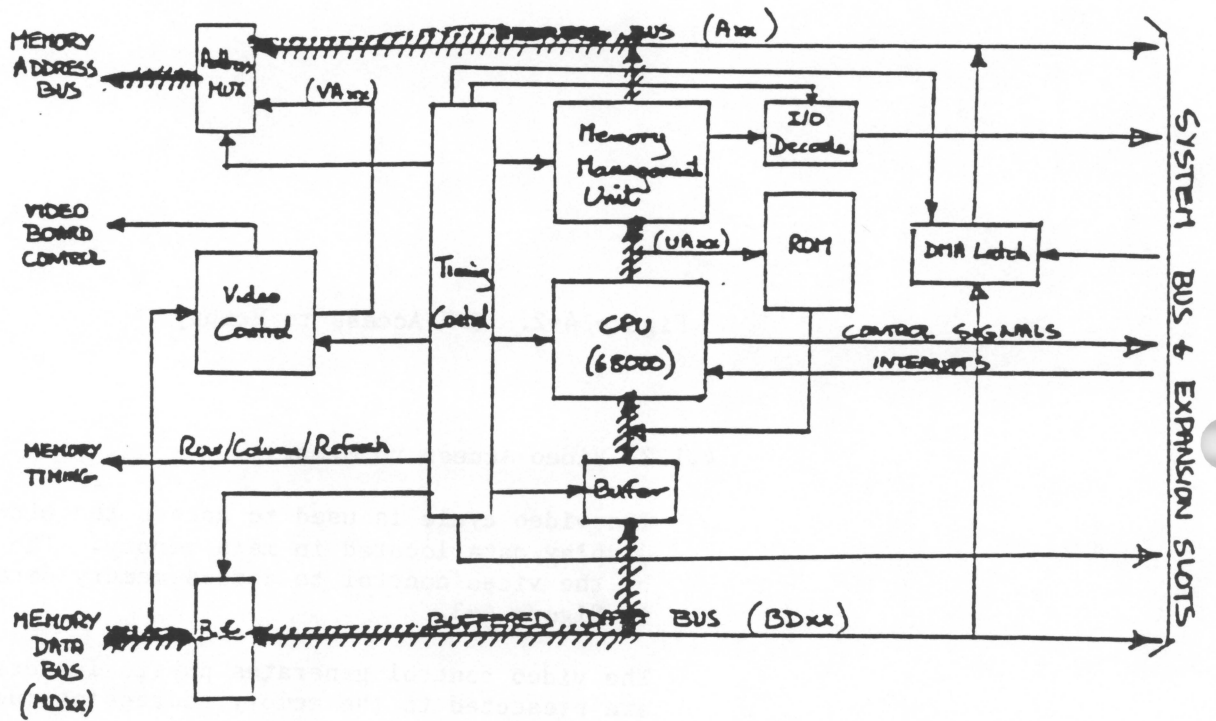


Figure 4-2. CPU Access to Memory

4-52  
4-46



Figure 4-3. Video Access of Memory

#### 4.1.3 CPU Access to I/O

The CPU communicates with I/O over the system bus. I/O can be a device either located on the I/O board, which is always resident, or can be located on an expansion board in one of the expansion slots.

The CPU generates the logical address of the I/O device with which it wishes to communicate and presents it to the MMU in a manner similar to a normal memory access.

Within the MMU, the SLR contains information that defines the segment being accessed as valid I/O space. This condition inhibits any memory access and instead presents the address generated by the MMU to the I/O decode logic.

The I/O decode logic generates an enabling signal to the board on which the device is resident, and the data transfer takes place via the system bus. Refer to Figure 4-4 for an overview of the paths taken.

Note that this cycle can be simultaneous with an access to memory by the video control, since no logic or data path is common to both.

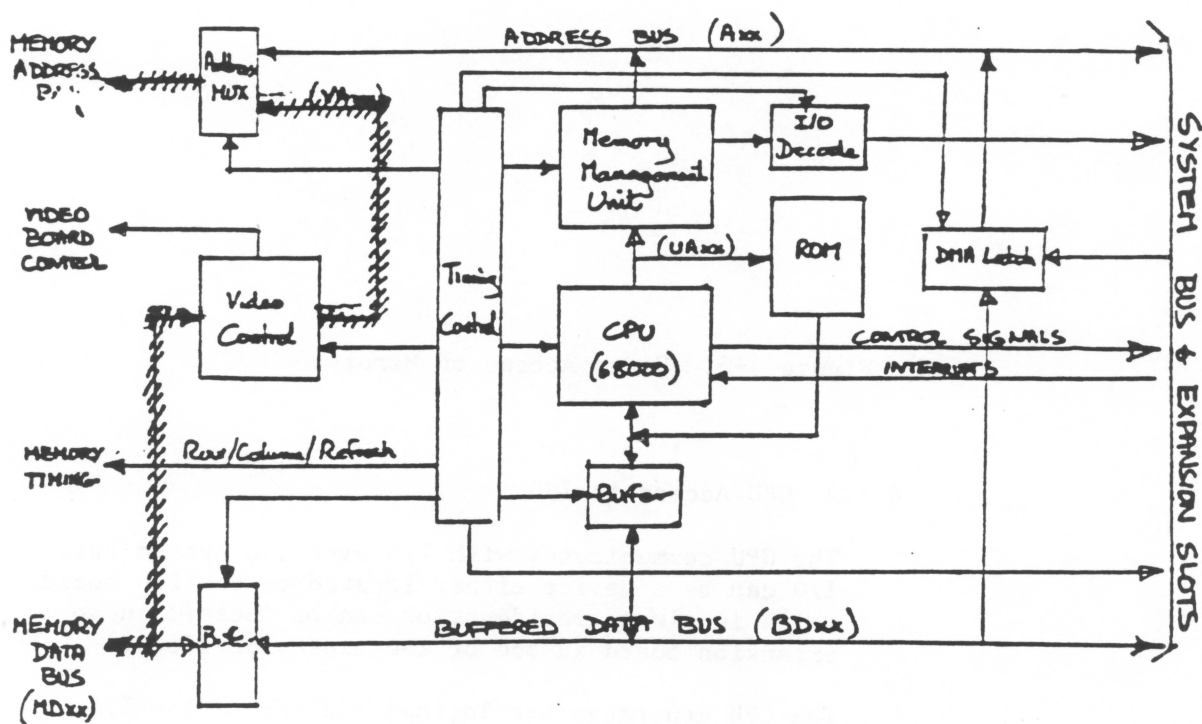


Figure 4-3. Video Access of Memory



Figure 4-4. CPU Access to I/O

#### 4.1.4 I/O Access to DMA Address Latch

Transfer of data between main memory and I/O devices in the Lisa can take place under DMA control. This relieves the CPU from involvement in the transfer of every byte to and from the peripheral.

In order to generate the required physical addresses and to present them to memory, the processor board is equipped with a latch to hold the high-order byte of the memory address. When the high-order byte is to be changed to allow access to another block, the DMA controller can load the latch as shown in Figure 4-5.

Note that this operation does not require use of the memory bus and can therefore proceed in parallel with a video access to memory.

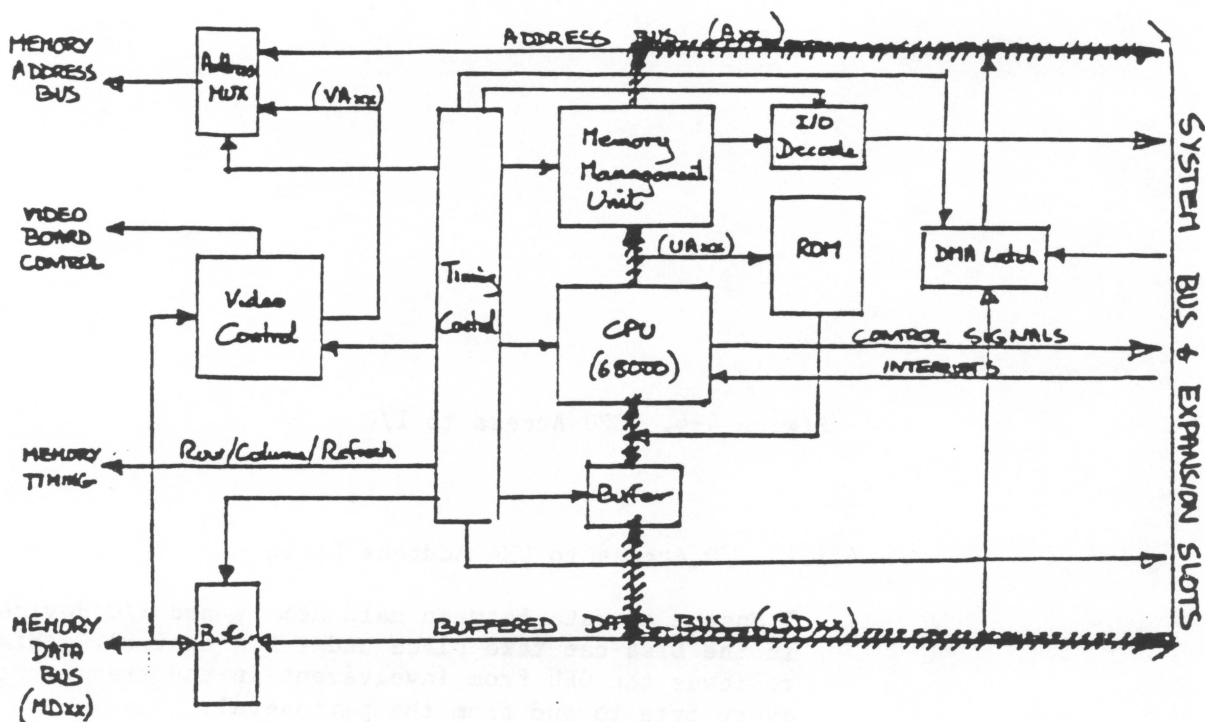


Figure 4-4. CPU Access to I/O

4-6b  
4-8



Figure 4-5. I/O Access to DMA Address Latch

#### 4.1.5 I/O Access to Memory (DMA)

As discussed in the previous subsection, the Lisa can use DMA to transfer data between memory and peripherals in a manner transparent to the CPU. The paths used to perform this are shown in Figure 4-6.







Figure 4-6. I/O Access to Memory (DMA)

A physical memory address is generated by the DMA controller from the DMA address latch contents for the high-order byte and address lines A1-A12 to give the full address.

This address is presented to the memory in a manner identical to a normal CPU access, and the data are transferred via the buffered data bus.

No other data transfer can take place during this portion of a DMA cycle. Each series of DMA cycles begins with a procedure in which the CPU relinquishes control of the bus. From this time, all signals on the bus that are required for data transfer are generated by the DMA controller.

It is important that the design of I/O cards that use DMA takes into consideration the overall timing and software constraints as described in section 4.2. Since the CPU is not processing instructions at this time, the I/O card "replaces" the CPU and must generate signal patterns that are indistinguishable from those of the CPU.

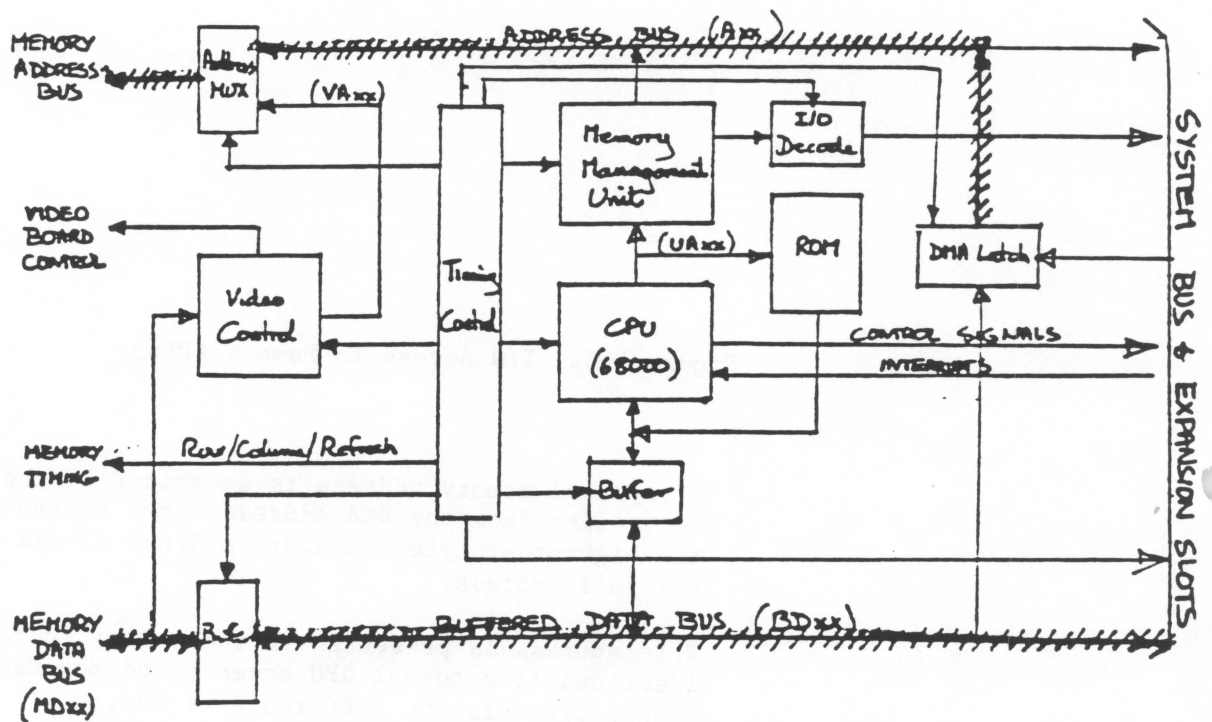


Figure 4-6. I/O Access to Memory (DMA)

## 4.2 Instruction Cycle and Timing

Apart from internal signals used on the board itself, the main signals generated by the processor board are used to control timing to the memory array, data refresh on the CRT screen via the video control, and data transfer on the system bus.

### 4.2.1 Internal Timing

The timing internal to the processor board is based on the timing of the 68000. The basic 800 ns 68000 bus cycle is divided into two equal halves. A 400 ns video cycle is followed by a 400 ns processor cycle.

A timing diagram of the principal signals internal to the processor board is shown in Figure 4-7.

Figure 4-7. Processor Board Internal Timing

Each half of the instruction cycle is in turn divided into eight timing periods  $t_0$  through  $t_7$ , each of 50 ns. These periods are generated by a 20.375 MHz signal called DOTCK. DOTCK is divided by four to provide the 5 MHz CPUCK signal used to clock the 68000. The processor board synchronizes to the 68000 so that the S- states of the 68000

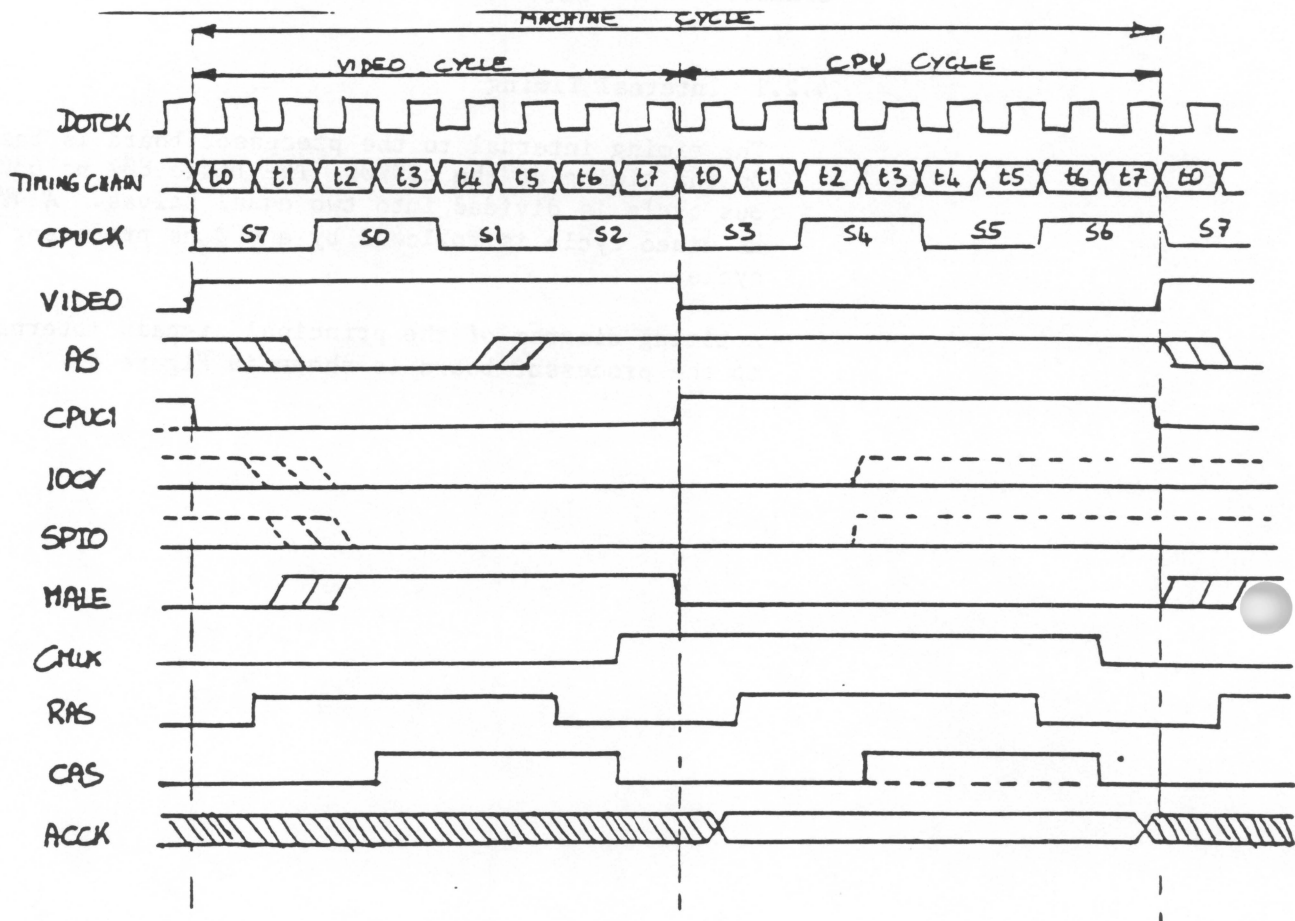


Figure 4-7. Processor Board Internal Timing

9b  
4-13e

correspond, as shown in Figure 4-7.

This timing results in the address strobe (AS) signal being recognized during the video cycle, which permits the MMU to calculate and check the physical address in time for the result to be ready for the processor cycle.

The CPUC1 signal signifies the first cycle of a CPU instruction execution. The IOCY and SPIO signals indicate that an I/O or special I/O cycle is in progress. The IOCY and SPIO signals are generated from information contained in the MMU.

The Memory Address Latch Enable (MALE) signal is used as the selector between the two registers in the MMU that are associated with each logical address. When MALE is true, the segment origin register (SOR) is being accessed and the MMU is in the process of calculating the origin of the page being addressed. When MALE goes false, the segment limit register (SLR) is selected and the MMU calculates whether the address lies within the page limits.

The CMUX signal is used to select the source of the address which is to be presented to the memory. When deasserted, it gates the video address. When asserted, it gates the physical address presented by either the MMU or the DMA control.

Memory timing is controlled by the row and column address strobes (RAS) and (CAS). These and all of the other signals listed above are generated with reference to one or more of the  $t_0$  through  $t_7$  time periods.

#### 4.2.2 Memory Management Timing

A logical address is being output from the 68000 when the address strobe signal is asserted. The MMU must process this and present the result to the memory on the basis of the type of access. In the case of a memory cycle, the physical address must be stable in time for presentation to the memory along with the row and column strobes.

In addition, the timing of each special I/O cycle with which data are written into an MMU register involves signals not used in any other operation. Figure 4-8 shows a sequence of operations as follows:



- \* The beginning of a relocate/check cycle
- \* The execution of a write to the MMU.

Figure 4-8. MMU Timing Diagram

The MALE signal selects between relocate mode, based on the SOR, and check mode, based on the SLR. MALE directly generates the base/limit (B/L/ ) selection signal.

The FC2 signal originates in the 68000. It is asserted to indicate that the processor is currently in supervisor mode. Mapping is forced to the supervisor context, context 0, when FC2 is asserted.

A write cycle to the MMU is a special I/O cycle and begins with the assertion of the SPI0 signal, which directly generates the MMUI0 signal. This causes

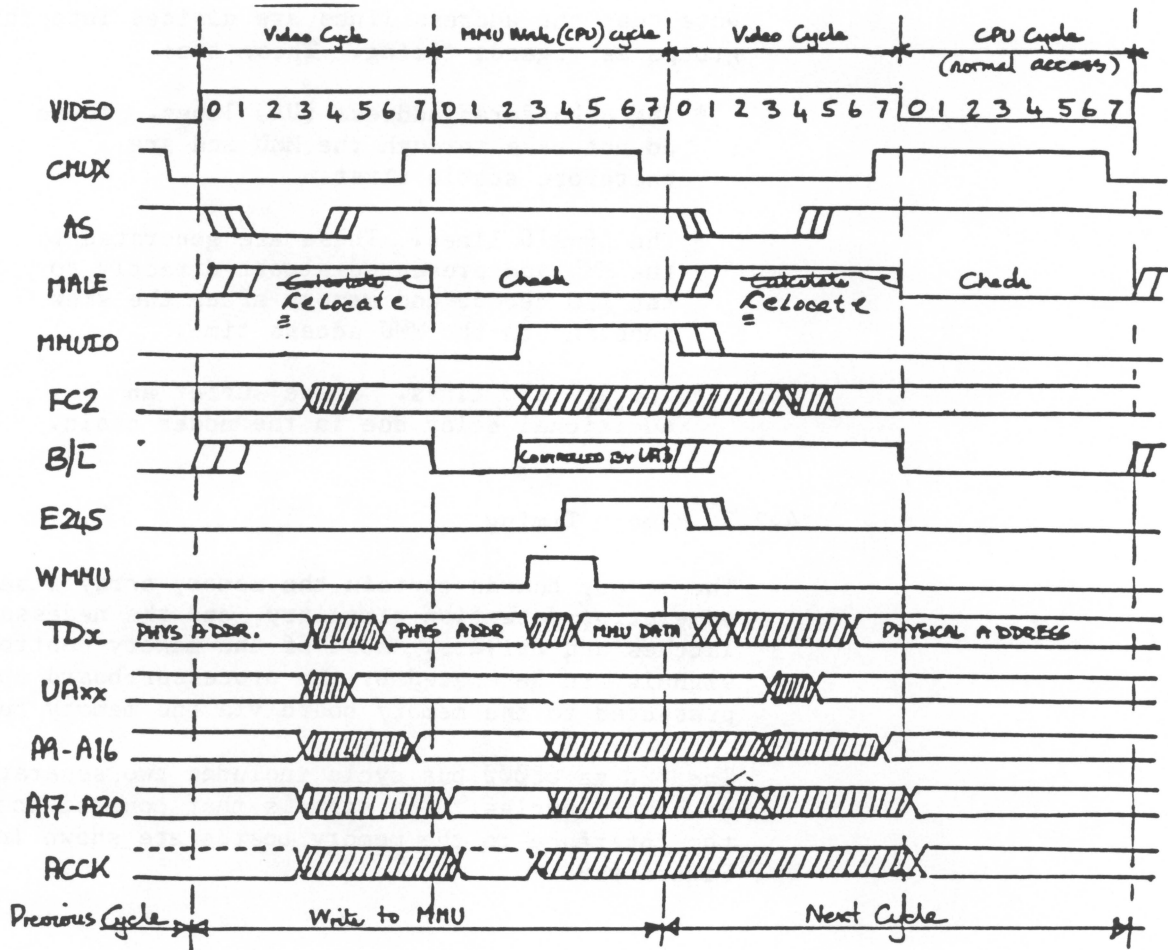


Figure 4-8. MMU Timing Diagram

the B/L/ signal to be controlled by the UA3 address line to select between the SOR and the SLR, and the SEG1 and SEG2 bits define the currently addressed context.

The output of the MMU memory is disabled when the WMMU signal is asserted and the data to be written is placed on the TD lines by means of the E245 signal.

Note that the address lines are divided into three groups as regards timing. These are:

- \* The unbuffered address (UA) lines. These do not pass through the MMU and are therefore stable first.
- \* The A9-A16 lines. These are generated by the MMU and presented almost directly to the I/O decode and memory after the skew inherent in the MMU access time.
- \* The A17-A20 lines. These suffer an additional delay due in the adder chain.

#### 4.2.3 Memory Timing

The memory boards contain the memory array itself, some error-detection circuitry, and the necessary latches and drivers. Most of the memory control signals are generated by the processor board and presented to the memory board via the memory bus.

The 800 ns 68000 bus cycle includes two separate 400 ns memory cycles. The signals that control these on the interface to the memory boards are shown in Figure 4-9.



Figure 4-9. Memory Control Timing Diagram

The CMUX signal is used to select between the video and the processor addresses being presented to the memory. The contents of the RAX signals are controlled by the CMUX and the RAS signals.

The A9-A16 and A17-A20 are generated by the MMU and are used to generate column address and board or device selection within memory. The RAS signal is present for every video cycle unless the cycle and once during every 68000 bus cycle.

The CAS signal is generated for each video cycle, but can be absent from a processor cycle if no memory access is taking place. This is indicated by the deassertion of CASEN.

The VA8-11 are representative of the timing of all video address signals. These are also used in the generation of the refresh address in the memory since video accesses are always to sequential locations. Refresh takes place during the video cycle.

Selection of refresh for a particular board is

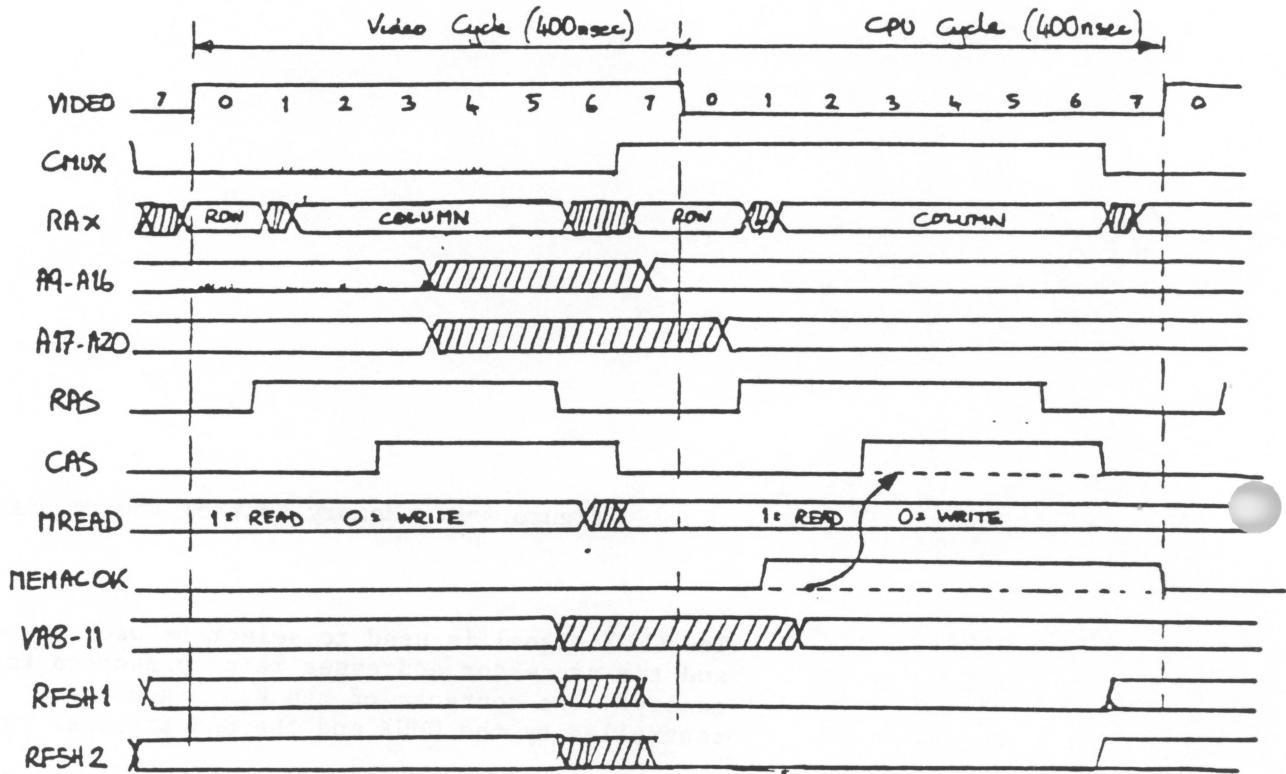


Figure 4-9. Memory Control Timing Diagram

controlled by the RFSH1 and RFSH2 signals. Details are discussed in section 5.3.

#### 4.2.4 Video Timing

The video board, which contains the analog circuits necessary to control the CRT, is described in Chapter 8. Timing signals for the transfer of data to the CRT are shown under processor board control. Figure 4-10 shows the timing diagram for the signals generated.

Figure 4-10. Video Control Timing Diagram

The video control contains a shift register that is loaded at the end of a video access and then shifted out at a rate of 1bit/50 ns, 20 MHz, to provide the bit-serial data stream to the video board.

The data stream continues to be read whenever the end of a line is reached and a horizontal retrace is being performed. However, the data are ignored and the video address counter is not incremented. In a similar fashion, data are not read out during the vertical retrace.



September 7th 1982

LISA Hardware Manual

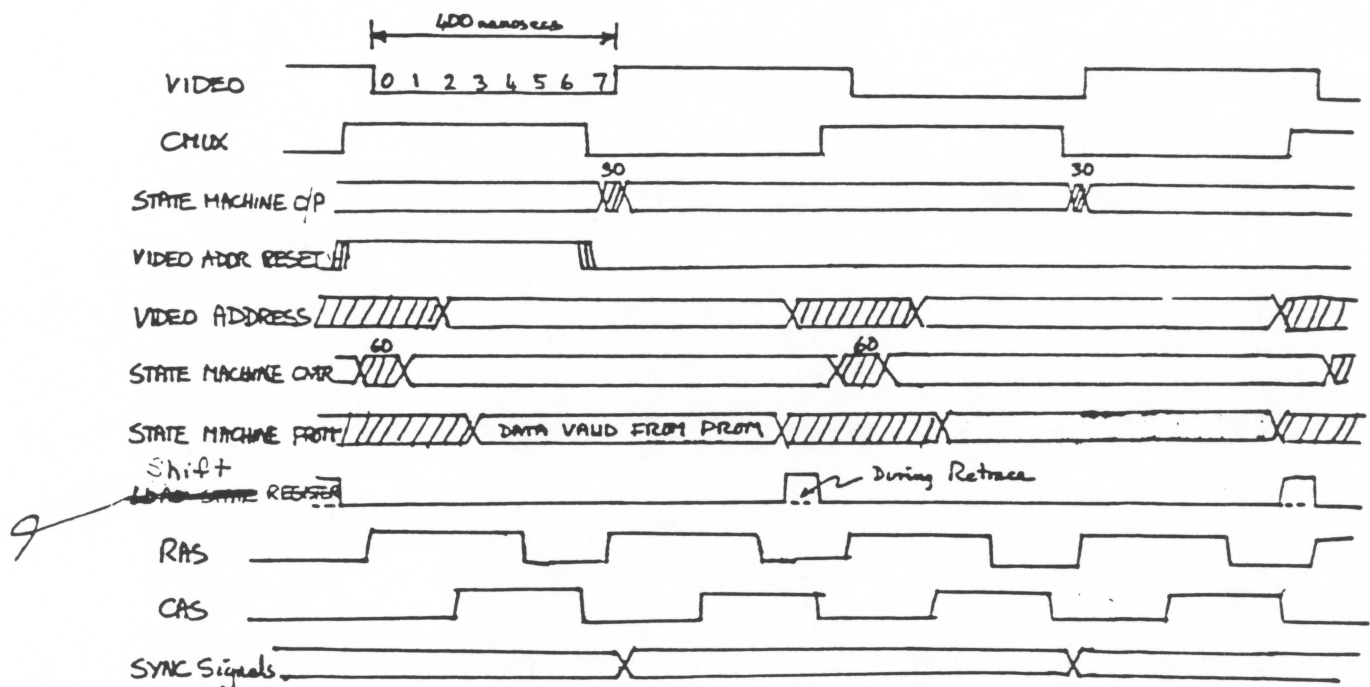


Figure 4-10. Video Control Timing Diagram

Timing for video control is performed by a state machine and is synchronized with the VIDEO signal. The shifting of the bits out to the video board is performed by the DOTCK.

The state machine counter is clocked with the falling edge of the VIDEO signal. The rising edge of this signal latches the current output from the state machine PROM.

The video address is reset once per page. This occurs after 379 lines of 720 pixels each. Only 364 lines are displayed on the screen. The timing of the video data within a page is shown in Figure 4-11.



Figure 4-11. Video Page Timing

The last pixel of each page must be a 1 (one) in order that the retrace is black. This is done by having a "one" in the least significant bit of the 32,767th byte displayed.

#### 4.3 The Central Processing Unit

The heart of the Lisa is the 68000 32/16-bit processor, which is described in detail in the 68000 User's

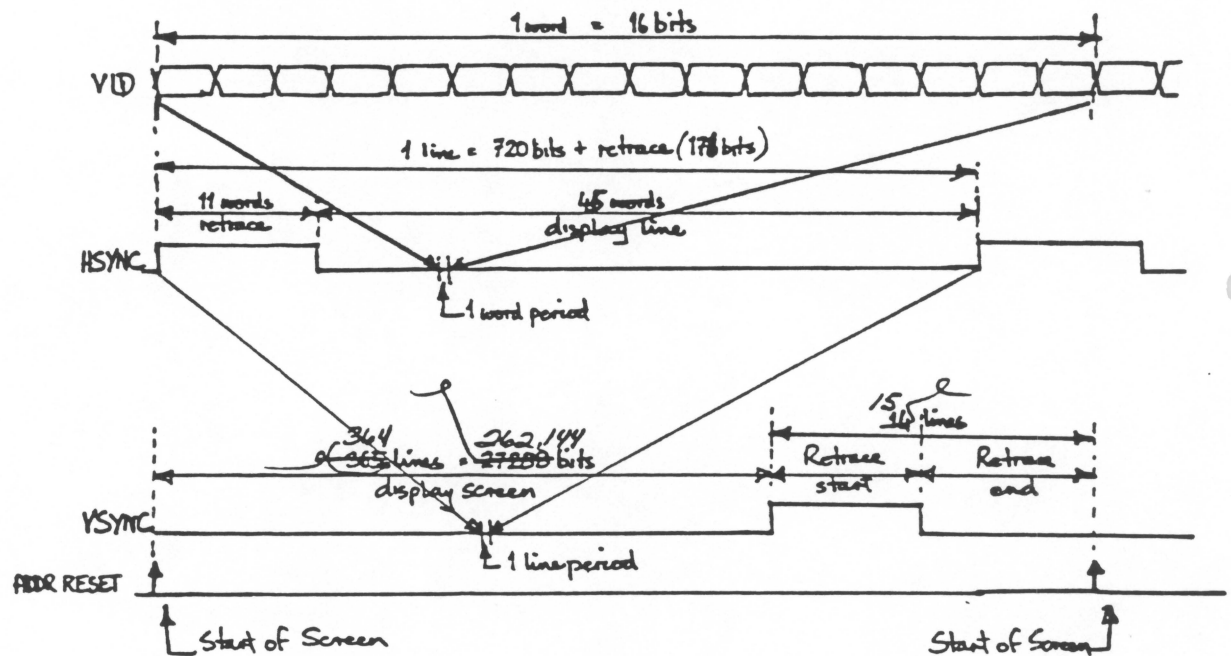


Figure 4-11. Video Page Timing

Manual. Refer to drawing 050-4009 in Appendix A for schematics of the devices discussed in this section. Page 3 of the schematics shows the CPU at location A3-D3.

#### 4.3.1 Clock Generation

Timing generation is shown on sheet 2. At B-4, crystal Y1 and its associated oscillator circuit are used to generate the DOTCK and CK signals. These both have a period of 50 ns and have the waveform shown on Figure 4-12.

The DOTCK signal is used to provide timing to the video control, which shifts bits serially to the video board. These bits are then displayed on the screen. CK drives the 4-bit synchronous counter at D4, sheet 2, whose outputs are decoded to provide the eight timing states  $t_0/$  through  $t_7/$ , plus the VIDEO signal, as shown in Figure 4-12.

The QB output has a period of 200 ns. It is used as the 68000 clock and also generates the main clock CPUCK, which is distributed through the Lisa.

Figure 4-12. Processor Board Timing Generation

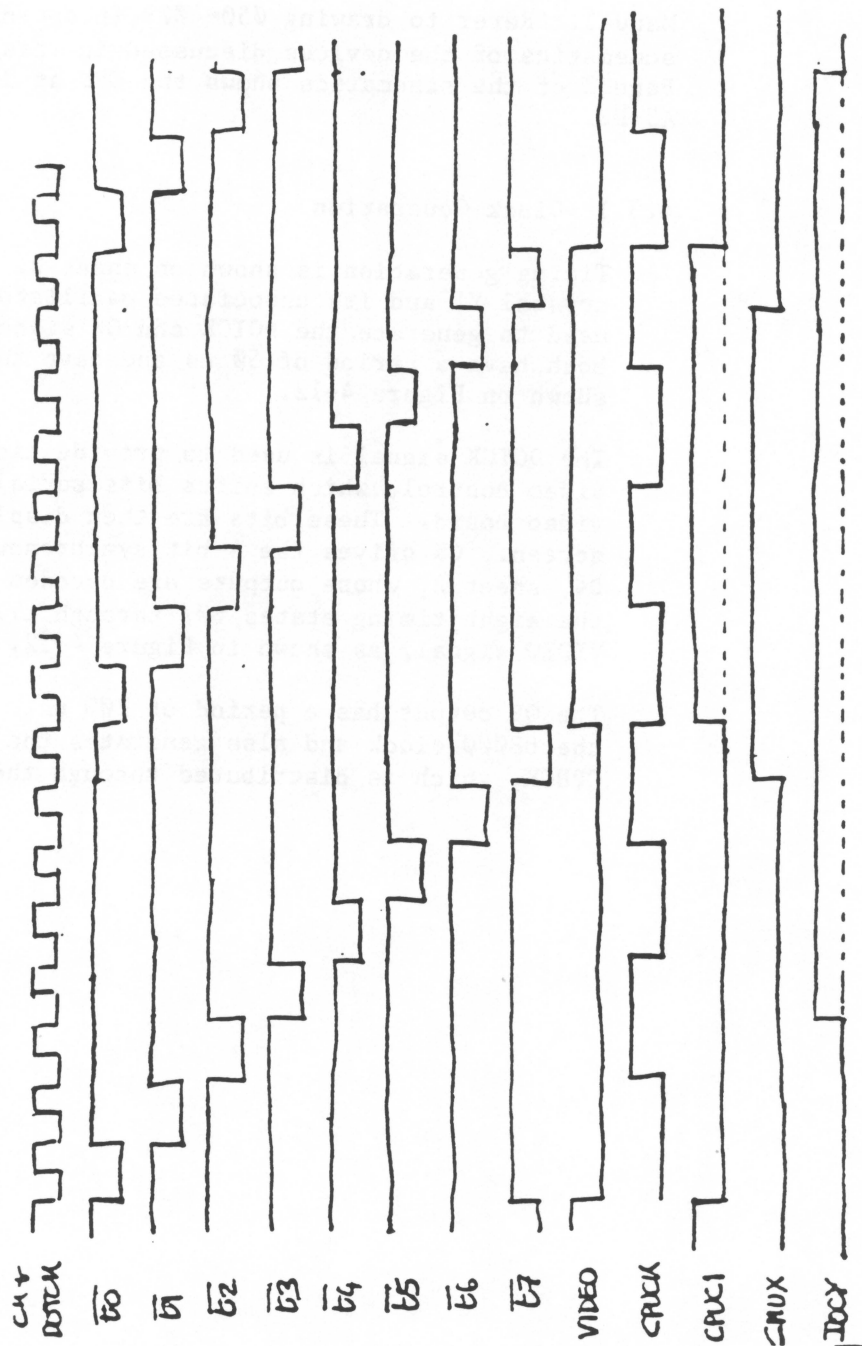


Figure 4-12. Processor Board Timing Generation

#### 4.3.2 Processor Control Signals

The six main bus control signals are buffered by the LS244 at C-3 on sheet 3. This is enabled when the bus grant acknowledge (BGACK/ ) signal has been deasserted. In other words, the bus is released at the same time that the bus grant acknowledge is generated, which allows the device to take control of the bus.

Note that UAS/ constitutes an unbuffered address strobe, which is not gated as above. The 68000 is synchronized with the AS signal. This signal should be used for synchronization on the board whether the bus is under CPU control or not.

The status signals FC0, FC1, and FC2 are decoded to provide the IAK and IAK/ signals. These latter signals are used to gate the address of the device that generated an interrupt. The FC2 signal is used to distinguish between supervisor and user mode cycles.

Incoming interrupts are encoded by priority through the LS148 at C-4 and presented to the IPL0-IPL2 inputs of the 68000. The system RESET is buffered and chained through the 7417 at D-3, whose output is pulled up through R12. Other control signals interface directly to the bus drivers, with no buffering.

#### 4.3.3 Address and Data Lines

Address lines UA9 to UA23 are presented directly to the MMU, since these are the bits that define the segment and page of memory that is being addressed. Refer to section 2.3 for a discussion of the MMU address translation process.

The eight low-order bits, A1 to A8, are buffered through the LS244 at A-3, sheet 3, which is gated with BGACK/ to release the memory bus when the CPU relinquishes bus control.

The 16 data lines, D0 to D15, are buffered by the bidirectional LS245s at A-4 and B-4. These are enabled by the DBON/ . The direction is selected by the READ signal.



#### 4.3.4 Bootstrap ROM

The bootstrap ROM is used during power-up to provide initial programs that permit the operating system to be loaded from a mass storage peripheral, and the Lisa itself, to be initialized to a known configuration.

The ROM is accessed by means of special I/O cycles. The ROM consists of the two devices shown at locations A-2 and B-2 on sheet 3.

Note that the ROM is addressed directly by the address lines from the 68000. This means that the ROM addressing does not require that the MMU be operational. Therefore bootstrap routines execute correctly out of ROM before the MMU has been configured at initialization time.

The ROM is enabled by the ROM/ signal. This signal is asserted whenever a special I/O cycle is in process while UA15 is high and UA16 is low.

#### 4.4 The Memory Management Unit (MMU)

The function of the MMU is to translate logical addresses used by the software into physical addresses. This is performed in terms of logical pages of 512 bytes within logical segments of 128 Kbytes.

The operation of the MMU is discussed in section 2.3 above. Address lines UA1-UA8 are not operated on by the MMU. UA17-UA23 are used to select one of the 128 possible logical segments, while UA9-UA16 address a page within that segment.

At the same time, the entire logical address space is in one of four possible contexts, selected by the SEG1 and SEG2 signals. The MMU has four identical sets of registers, one for each context of 128 segments.

Each segment has two registers associated with it. The SOR contains 12 bits that give the page number with that the segment begins. The SLR contains 8 bits that supply the number of physical pages assigned to this segment, plus 4 bits that indicate the physical address space and other data about the physical segment.

The MMU is shown in simplified form in Figure 4-13. It is used to define three distinct physical address spaces. This is discussed in Chapter 2 in some detail. Refer also to Figure 2-4. The 68000 can access a maximum of 2 Mbytes of RAM, which is located on up to two boards, plus a distinct I/O space and an additional

special I/O space. Each address space is a 2 Mbyte block.

The MMU logic is shown on sheet 4 of schematic 050-4009 in Appendix A. It consists of RAM storage for the segment registers, address latches, and logic that permits the contents of the registers to be manipulated.

Figure 4-13. MMU Block Diagram

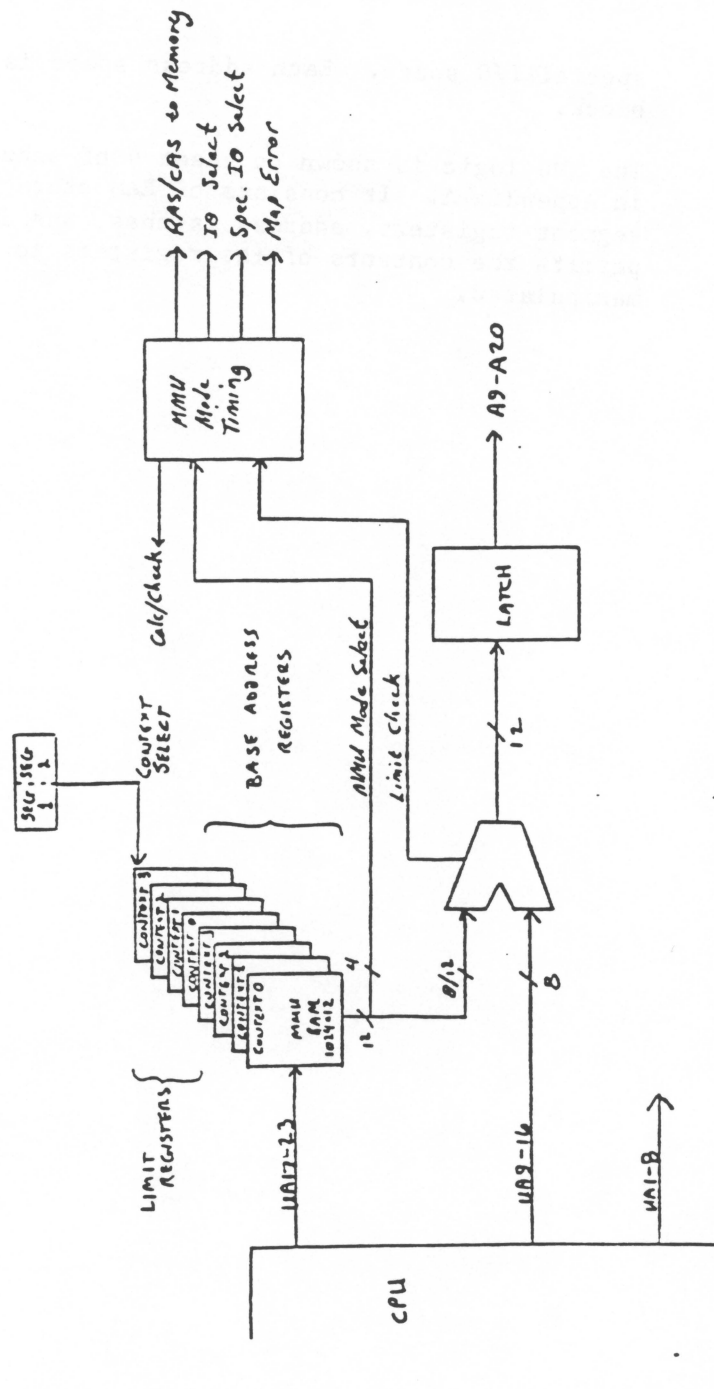


Figure 4-13. MMU Block Diagram

196  
4-26e

#### 4.4.1 MMU RAM Storage

The registers of the MMU are organized within a 1K by 12 bit RAM memory matrix. These can be altered by software when the 68000 is in any mode. The operating system is responsible for controlling access to these registers. The register addresses are generated from the logical address. Their contents define the physical address.

The Lisa has a 16 Mbyte logical address space because of the architecture of the 68000. This entire space can function in one of four contexts, depending on the configuration of the SEG lines.

Each 16 Mbyte space is divided logically into 128 segments, each of which is allocated a pair of registers in the MMU that define segment parameters. This results in a total of 1024 registers, since the four contexts each contain 128 segments and each segment requires a pair of registers. This is shown in Figure 4-14.

<hr/>		
Segment 0 Base (SOR)		^
Segment 0 Limit (SLR)		
Segment 1 Base		
Segment 1 Limit		
Segment 2 Base		
Segment 2 Limit		
.	Context 0	
.		
128 pairs total		
.		
.		
Segment 127 Base		
Segment 127 Limit		v
<hr/>		
Segment 0 Base		^
Segment 0 Limit		
.		
.		
128 pairs total	Context 1	
.		
.		
Segment 127 Base		
Segment 127 Limit		v
<hr/>		
Segment 0 Base		^
.		
.		
128 pairs total	Context 2	
.		
.		
Segment 127 Limit		v
<hr/>		
Segment 0 Base		^
.		
.		
128 pairs total	Context 3	
.		
.		
Segment 127 Limit		v
<hr/>		

Figure 4-14. MMU Memory Configuration

The two registers for each segment are known as the SOR and the SLR, as described in Chapter 2.

The SOR defines the physical address of the first

byte of the segment. This is given as a multiple of 512-byte blocks. The first block in memory is thus defined by address 000000@16.

The SLR defines the size of the segment in terms of multiples of 512-byte pages. It also contains a 4-bit code that defines the type of space, as outlined in Figure 2-4.

The MMU memory is thus logically divided into four sets of 128 pairs of 12-bit registers, each set being used only when the 68000 is running in the corresponding context. This enables software to perform context switching very rapidly.

Referring to schematic 050-4009, page 4, the ten lines used to address the MMU RAM can be seen as consisting of the following:

- \* UA17-UA23, 7 address lines,
- \* MS1 & MS2, from SEG1 & SEG2 lines,
- \* B/L/, the base/limit select.

The address lines are the unbuffered high-order address lines from the 68000 CPU on sheet 3. The other signals are generated in the logic at C-4. Refer to Figure 4-8 for the timing relationships.

The crucial element of operation in the MMU is that it must first generate the physical address that is the origin of the memory page to be accessed. Then it must check whether the address lies within the limits of the page addressed.

These two functions are selected by the memory address latch enable (MALE) signal, which originates in a S109 JK flop on sheet 2. MALE is asserted at the end of a CPU cycle when the AS signal becomes deasserted. This typically occurs around the t3 period of the video cycle. This asserts the B/L/ signal through the LS00 gate at C4, on sheet 4, which selects the base register.

The SEG1 and SEG2 latch is shown on sheet 5. The software defines the context in which the CPU is currently running by means of these two signals, except if the CPU is running in supervisor mode. Supervisor mode is indicated by the FC2 signal being asserted. The segment being addressed is defined by the UA17-UA23 signals.

When the conditions exist to set the CPUC1 JK flop at B-2 on sheet 2, the same term is used to reset

the MALE flop at A-2. This occurs at the end of  $t_7$  time during the video cycle. With MALE deasserted, the B/L/ term also becomes deasserted and the SLR that corresponds to the SOR just accessed is used to check the access limits and also give the type of cycle to be performed.

#### 4.4.2 SOR and SLR Initialization

The MMU registers are in an undefined state when the Lisa is first powered on. The bootstrap ROM is not addressed via the MMU. This enables the CPU to operate initially without requiring that the MMU be functional.

Before RAM can be used, the registers that correspond to the contexts and segments being used must be initialized. The RAM that is used as the MMU register storage is accessed via special I/O space. This is decoded by the LS139 device at D-2 on sheet 5. The MMUIO/ signal enables the WMMU/ signal via the JK flop at D-4, on sheet 4. Refer to Figure 4-9 for signal timing relationships.

Each register is written to in a separate write cycle. The SEG1 and SEG2 lines are configured to the appropriate context and the UA17-UA23 lines select the segment. When the registers are written, the SOR or SLR are selected by the state of the UA3 line through the LS00 gate at C-4 on sheet 4.

The MMUIO signal is asserted when all the following conditions exist:

1. The SPIO signal is asserted
2. UA15 is asserted
3. UA16 is deasserted.

When MMUIO becomes asserted, the clear input of both S109 JK flops at D-4 are released, allowing the E245 signal to be asserted at  $t_4$  time and WMMU/ to be deasserted at  $t_3$  time if the 68000 is performing a read cycle.

The data to be written into the registers is presented to the MMU RAM via the two LS245 bidirectional drivers at C-3 and D-3. The direction in which the data are moved is controlled by the READ signal from the CPU.

The TD lines from the lower LS245 write the two low-order nibbles into the lower RAMs. The upper



nibble is base data for the SOR or access-type data for the SLR. Register contents can be read via the same path.

#### 4.4.3 Address Translation

During the video half of the 68000 bus cycle, the SOR contents are read out as a 12-bit physical address that defines the physical address of the beginning of the segment at a 512-byte boundary.

This value is added to the page address that is presented to the other inputs of the adder by UA9-UA16. Note that the high-order input nibble is forced to zero. The result is presented to the LS373 latch at C-2 and the LS374 latch at B-2. On the falling edge of the MALE signal, which occurs at the end of the calculation involving SOR, the physical address is latched. This will be stable on the A9-A20 lines if the BGACK/ signal is deasserted, no DMA access in process, and CMUX is asserted to avoid conflict with the video address.

MALE being deasserted signals the second half of the 68000 bus cycle, when the access limits are checked. The B/L/ signal changes polarity and the contents of the SLR are read out.

The eight low-order bits, which indicate the number of pages contained in this segment, are also added to the page address given by the UA9-UA16 lines from the CPU. The overflow line from the second nibble is the access check (ACCK) signal. If ACCK is asserted, this indicates that the desired page lies outside the limit set for the segment in question.

An exception to this occurs when the control bits indicate that the accessed segment is a stack segment. Since the stack begins at the high-order address within the block, the significance of ACCK is inverted in this case. This function is implemented in the S86 gate at B-3 on sheet 2.

An overflow results in the suppression of the CAS/ signal, which prevents any memory operation from taking place. The output of the high-order nibble of the adder is ignored.

The high-order nibble of the SLR contents provide flags for the type of segment that is being accessed. These indicate:

\* Segment in memory space (MEM)

- \* Segment in I/O space (IO)
- \* Segment that is read-only (RO)
- \* Segment that contains the stack (STK).

Refer to Figure 2-4 for the full coding permutations possible, since all combinations are neither valid nor covered by the above. These signals in turn generate the appropriate control signals for the type of segment indicated.

The MEM term being asserted, MEM/ low, enables the CAS signal for a memory access via the LS02 gate at C-3 on sheet 2.

The IO term being asserted initiates an I/O cycle via the IOCY flop at B-1 on sheet 2.

The RO term being asserted inhibits a write cycle to memory via the ALS32 gate at B-3 on sheet 2. It also inhibits an SPI0 cycle by being one of the terms on the LS260 gate at C-1.

The STK term being asserted causes a carry input to be presented to the low-order adder via the F02 gate at A-3 on sheet 4. This is done because of the stack configuration, which begins at the top of the segment and decrements through memory from there.

#### 4.4.4 Memory Timing Generation

Memory control timing is shown in Figure 4-9. Since the nine low-order bits of the address are presented directly from the CPU, these are used as memory row addresses. This avoids the need to wait for the output of the MMU to become stable before any memory addressing can be done.

The RAS is enabled through the LS32 OR gate at D-4 on sheet 2 of the schematics. RAS goes true at the end of the  $t_0$  state and goes false at the end of  $t_5$ . Note that RAS is generated even for cycles that turn out to be I/O or erroneous. This does no harm provided that CAS is generated for such cycles. The only time that RAS is generated for a memory cycle is if the current cycle is a video cycle and is not the first of a multi-cycle CPU access.

The CAS has an enable signal that is the logical OR of a number of terms:

- \* If a video cycle is in process

- \* If a DMA cycle is in process (BGACK!.CPUC1)
- \* If no error was detected in a memory cycle.

This last term requires that a number of factors be satisfied:

- \* A memory cycle is in process (MEM!.CPUC1),
- \* No attempt is made to write to a read-only segment (READ/!.RO),
- \* The segment limit was not exceeded (ACCK).

If CAS is enabled, it will go true at the end of t2 and goes false at the end of t6.

The signal requesting a read from the memory is MREAD. It is generated through the S02 gate at D-2 on sheet 5. Refresh of the memory is generated by two gates at A-4, sheet 2. One of the two signals R1 and R2 is provided to the memory board for use as a refresh enable. Each occurs when either, but not both, of the VA8 and VAl1 video address signals is asserted.

#### 4.5 Video Control

The video control is shown in Figure 4-15 and consists of the following components:

- \* Video address counter, VA1-VA14,
- \* Shift register
- \* Video state machine
- \* Video page register, VA15-VA20.

The 32 Kbytes of memory in which a full video screen is stored is sequentially accessed once every 1/60th of a second.



Figure 4-15. Video Control Block Diagram

#### 4.5.1 Video Address Counter

The video address counter points to the location in the 32 Kbyte video page of the next data byte to be fed to the display. It is incremented once for every 16 active pixels that are fed into the video board, and it is reset at the end of each vertical retrace.

The circuitry is shown on sheet 5 of schematic 050-4009. It consists of the two LS393 counters at B-4. The counters are arranged in series and the outputs fed to the memory address MUX in the upper right of sheet 4. The counter is clocked on the leading edge of each t7 state via the JK flop at B-3, sheet 5. It is reset synchronously with CMUX when the video state machine determines that the vertical retrace has been completed.

#### 4.5.2 Video Data Shift Register

This is used to convert the 16-bit words of video data read out of memory into the serial data stream required by the video board.

The circuit is also shown on sheet 5 of the schematics. It consists of the two LS166 devices at

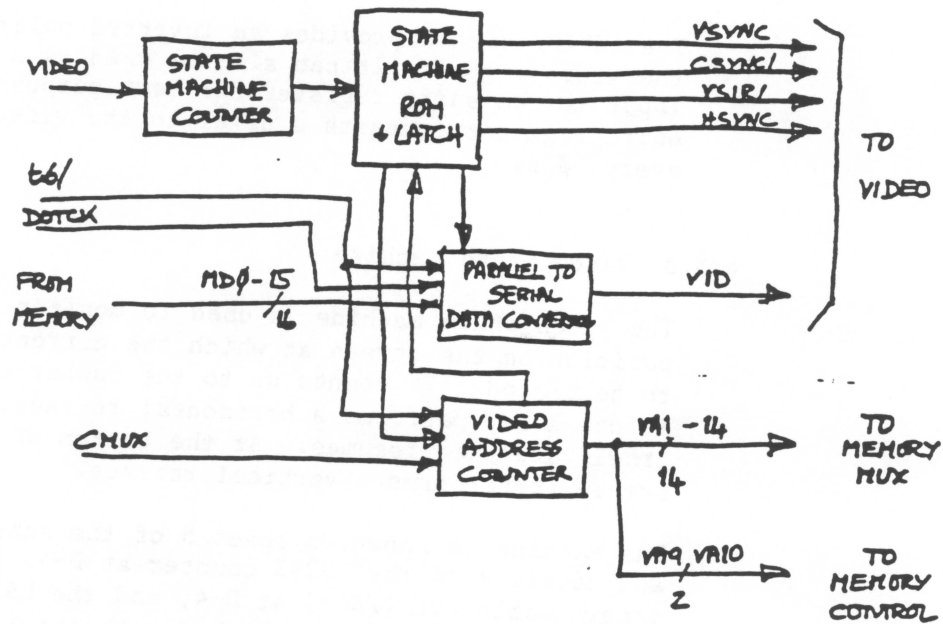


Figure 4-15. Video Control Block Diagram

B-3 and some associated circuitry. The clock that increments the video address counter is also used to parallel load the shift register with data presented on MD0-MD15.

The DOTCK signal from C-4 on sheet 2 is used to shift the data out to the VID output via the JK flop at C-3 on sheet 5. This flop has been inserted in the data path to ensure a uniform hold length for each data bit. If this were not present, the final bit in each word could be curtailed, since the next word is parallel-loaded into the shift register.

The INVID/ signal provides an inverted polarity of the video signal. It can also be used as a serial input to the shift register for test purposes. The shift register presents a pixel to the video board every 50 ns.

#### 4.5.3 Video State Machine

The video state machine is used to monitor the position on the screen at which the current data are to be placed. It counts up to the number of words in one line, performs a horizontal retrace, resets itself and then resumes. At the bottom of the screen it performs a vertical retrace.

The machine is shown on sheet 5 of the schematics and consists of the LS393 counter at D-4, the 6309-1 programmable ROM (PROM) at D-4, and the LS374 latch at D-3. The counter is clocked with the video signal. The video does access memory during the video retrace periods. Both the counter outputs and the VA9 and VA15 signals are used to address the PROM. The data output from the PROM is latched in the LS374 with the opposite edge of the same signal that clocks the counter.

The HSYNC/ and VSYNC/ signals are presented to the video board for use in horizontal and vertical synchronization of the data on the screen. Other outputs are used to clear the video address counter at the end of vertical retrace, generate the VSIR/ to interrupt the CPU during vertical retrace for cursor positioning and reset the shift register load flop and the state machine counter itself.

The state machine performs 45 normal word fetches and shifts in each line before performing a horizontal retrace and resetting itself. When it detects the 364th line by monitoring VA15, it generates a vertical sync of 6 lines duration,

followed by vertical sync termination of an additional 9 lines duration.

#### 4.5.4 Video Page Register

The Video Page Register contains the six high-order bits of the address to be presented to memory, which are not provided by the Video Address Counter (A15-A20). This provides a physical address to the memory, that is, it is not translated by the MMU.

The circuit consists of the LS374 at B1 on sheet 4. It is loaded by the VAL/ signal, which is decoded from an I/O command at B-2 on sheet 5. It is gated as an address whenever the CMUX signal is false.

#### 4.6 Bus Interfaces

The processor board interfaces to three other components inside the cabinet. These are the:

- \* System bus
- \* Memory board(s)
- \* Video board.

The bus interfaces to all three are made by a single connector to the motherboard and are shown schematically in Figure 4-16.

The system bus on the motherboard is extended in part to become the expansion bus. This bus is effectively a subset of the system bus used by the I/O board. It is described in detail in section 3.4 of this manual.





Figure 4-16. Processor Board Bus Interfaces

#### 4.6.1 System Bus Interface

The system bus operates under control of the processor board during most normal operations. An exception occurs when a peripheral controller performs data transfer to or from memory by means of a DMA operation.

Refer to Figure 4-16 for a signal list and pin assignment. An overview is also shown on sheet 1 of schematic 050-4009.

The system bus interface signals can be divided into the following categories:

- \* Address lines

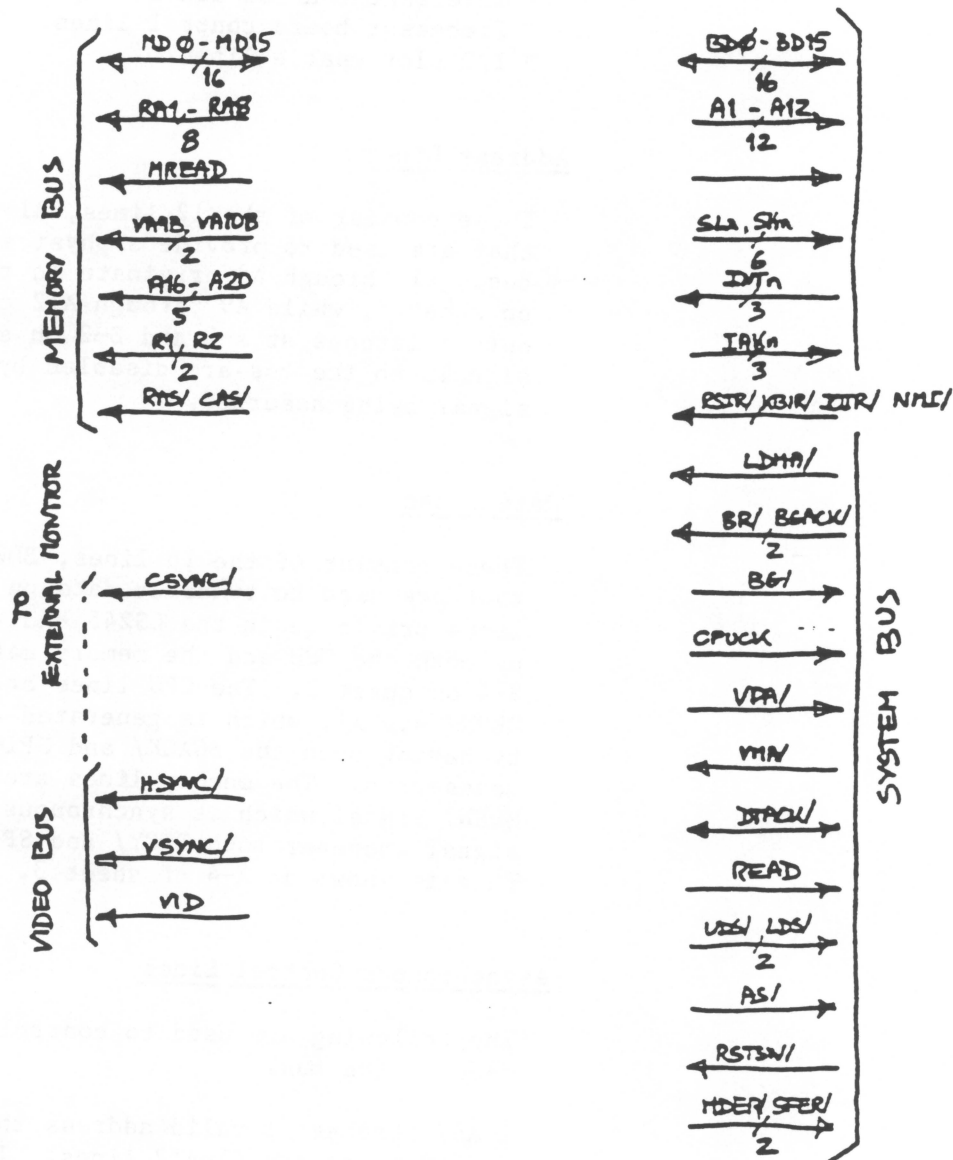


Figure 4-16. Processor Board Bus Interfaces

- \* Data lines
- \* Asynchronous control lines
- \* Bus arbitration lines
- \* Interrupt control lines
- \* Processor board control lines
- \* I/O slot enable lines.

### Address Lines

These consist of the 12 lines, A1 through A12, that are used to provide a physical address on the bus. A1 through A8 originate in the LS244 at A-3 on sheet 3, while A9 through A12 come from the MMU output latches at A-2 and B-2 on sheet 4. All signals to the bus are disabled by the BGACK/ signal being asserted.

### Data Lines

These consist of the 16 lines, BD0 through BD16, that are used to transfer data on the bus. All 16 lines originate in the LS245 bidirectional drivers of both the CPU and the memory matrix at A-4 and B-4 on sheet 3. The CPU lines are enabled by the DBON/ signal, which is generated at B-2 on sheet 5 by having both the BGACK/ and SPIO/ signals deasserted. The memory lines are enabled by the MDEN/ signal which is synchronous with the CPUC1 signal whenever both IOCY/ and SPIO/ are false. This is shown at A-4 of sheet 3.

### Asynchronous Control Lines

The following are used to control data transfers made on the bus.

AS/ strobes: a valid address that is present on the A1-A12 lines. It originates at C-3 on sheet 3.

READ: defines the cycle to be from memory to the accessing peripheral or processor. It originates at C-3 on sheet 3.

UDS/ and LDS/ : the upper and lower data strobes that define on which half of the BD0-BD15 lines a byte transfer is being made. They originate at C-3 on sheet 3.

DTACK/ : a data transfer acknowledge that indicates to the current bus master that

the asynchronous operation has been completed.

CPUCK: the clock used for the CPU itself. It is presented on the bus for synchronization purposes.

#### Bus Arbitration Lines

These signals are used to determine which device can operate as a bus master.

BR/ : the bus request line that is asserted by a slave device that wishes to have control of the bus.

BG/ : the grant signal with which the current master allows the requesting slave to take control of the bus. The signal originates directly on the CPU at C-3 on sheet 3.

BGACK/ : the signal with which the slave acknowledges that it has accepted the bus grant and has taken control of the bus.

#### Interrupt Control Lines

The following are used to determine the source of an interrupt and to acknowledge that the interrupt is being processed.

NMI/, RSIR/, INT0/, INT1/, INT2/, KBIR/, and IOIR/: signals presented to the processor board as an interrupt request.

IAK0/-IAK2/ : the coded responses to an interrupt request that indicates which requesting device is being serviced.

#### Processor Board Control Lines

These signals are used for general control and information within the Lisa.

RESET/ : a signal to the Lisa to return to its original on state.

LDMA/ : the signal to load the upper DMA address latch, which is located at D-1 on sheet 3 of the schematics. This signal

originates in the peripheral performing the DMA which needs to have the DMA upper address altered.

E: the enable signal for use with 68000-type peripheral devices.

VPA/ : an indication that a 68000-type peripheral is attached to the bus and has been addressed.

VMA/ : the response of the CPU to a VPA/ signal and notifies the peripheral that a valid address is present on the bus.

#### Slot Enable Lines

These lines decode the addresses of I/O slots that are not present on the processor board. They are presented to the bus as enable signals.

SL0/ -SL2/ : indicate that a slot low decode has been made for an I/O transfer to or from one of the three possible expansion boards. The signals originate in the decode logic at B-2 on sheet 5.

SH0/ -SH2/ : indicate the same for a high decode.

INTIO/ : indicates a decode of the processor board I/O space.

#### 4.6.2 Memory Bus Interface

The memory bus is used to provide control signals to the memory boards installed in the Lisa and also to perform transfers of data between the memory matrix and the rest of the memory. The signals are located on the same common connector to the motherboard as the system bus interface.

The signals that compose the memory bus are described below.

A16-A20: address lines used for internal decoding within the memory matrix and between installed memory boards. Refer to Chapter 5 for details.

RA1-RA8: the multiplexed address lines used to address the dynamic RAM array in conjunction with the RAS/ and CAS/ timing signals.

MD0-MD15: the data lines across which a word of data is transferred to and from the memory array. They originate at A-4 to B-4 on sheet 3.

MREAD: the memory read signal that indicates the direction of transfer. It is generated from the coincidence of the CPUC1/ and READ terms at D-2 on sheet 5.

RAS/ and CAS/ : the row and column strobes that indicate the contents of the RA1-RA8 lines. They are generated at D-1 and C-1 on sheet 2.

HDER/ : indicates a hard memory error, which is an error found to be uncorrectable if any Error Correction Code (ECC) is present in the memory. It is received by B-2 on sheet 3.

SFER/ : indicates a soft memory error, which is an error in memory that could be corrected by any ECC present in the memory.

VA9B and VAL0B : video address lines presented to the memory board for use in row refreshing.

R1 and R2 : signals that indicate which of the two possible memory boards is to be refreshed.

#### 4.6.3 The Video Interface

The Video Interface also shares the main connector to the motherboard and consists of the signal group listed below.

HSYNC/ : the horizontal synchronization pulse used to indicate to the video circuit that the end of a line has been reached. It originates in the output of the video state machine and is shown at C-3 on sheet 5.

VSYN/ : the vertical synchronization pulse used to indicate to the video board that the bottom of the screen has been reached. It originates in the video state machine at C-3

on sheet 5.

CSYNC/ : a composite of the above two signals and is used as an output to an external video monitor. It originates in the video state machine at C-3 on sheet 5.

VID : the data bit-stream output to the video board. It originates at C-3 on sheet 5.

#### 4.7 Decode and Latches

This section includes the miscellaneous circuitry present on the processor board but not included in the preceding. It can be considered under the following headings:

- \* I/O decode
- \* Processor board control register
- \* Memory error latch
- \* Processor board status latch
- \* Time delay logic.

##### 4.7.1 I/O Decode

This is located at the extreme right of sheet 5 of schematic 050-4009. The first stage consists of the LS138 1-of-8 decode device at A-2. This is enabled when an I/O cycle is in process via the IOCY term and the JK flop at A-2. The A16/ term must be asserted.

The resulting decode of the A13-A15 terms results in the eight decodes for the ranges shown. All but the high-order decode are presented to the memory on the system bus as described in subsection 4.6.1 above. The final term is CPUIO/ , which is used to enable the processor board I/O decode performed by the LS139 decoder at B-2 on sheet 5.

The A11 and A12 terms are decoded to give the address block decodes as follows:

RBES/ (F800-FFFF), the Read Bus Error Status, which is used to gate the processor board status latch at B-2 on sheet 3 onto the data bus.

RMEA/ (F000-F7FF), the Read Memory Error Address signal, which gates the latched address at D-1 and D-2 on sheet 3 onto the



data bus.

VAL/ (E800-EFFF), the Video Address Latch signal, which is used to load the upper 6 bits of the video address (A15-A20) at B-2 on sheet 4.

SYSC/ (E000-E7FF), the Processor Board Control Register signal, which is used to load the LS259 latched decoder at D-2 on sheet 5. Refer to subsection 4.7.2.

Decode during special I/O is done by the other half of the LS139 at B-2. It is enabled by the SPIO/ term and decodes UA15 and UA16 to give the ROM/ or MMUIO/ signals. These respectively enable the boot ROM on sheet 3 and allow data in the MMU registers to be modified or read.

#### 4.7.2 Processor Board Control Register

The processor board control register is shown at D-2 on sheet 5. It is used to hold control signals used internally on the board. It is loaded by the SYSC/ signal as described in the previous subsection. The signals latched have the following functions:

DIAG1/ and DIAG2/ : used to force a soft and a hard memory error respectively for diagnostic purposes. They do this when a write operation is performed to memory. This can be seen by the gating terms at D-2 on sheet 5.

SEG1 and SEG2: used by the software to select the context in the MMU in which the address translation takes place. They are input to the MMU RAM addressing logic at C-4 on sheet 4. Functional details are discussed in section 2.3.

START/ : used to disable the start mode. It is used when the program is to execute out of ROM after a power-on or reset. It enables access to the MMU RAM via the LS32 gate at B-4 on sheet 4.

SFMSK/ : used to suppress the detection of a soft memory error. It is applied to the LS279 Quad RS device at C-2 on sheet 3 to inhibit detection of the SFER/ signal.

VTMSK/ : used to suppress the vertical retrace interrupt to the CPU. It is input to the interrupt latch at B-2 on sheet 3.

HDMSK/ : similar to SFMSK/ above but suppresses the detection of a hard memory error. It is input to the latch at B-2 on sheet 3.

#### 4.7.3 Memory Error Address Latch

When either a hard or a soft memory error occurs, the processor board makes the address at which the error occurred available to software.

The latch consists of two LS374 octal latches, shown at D-1 and D-2 on sheet 3. The address is updated each memory cycle by using the CAS signal from sheet 2 as the gating term to the LS11 gate at C-1, which provides the clock to the latches. The occurrence of either type of memory error prevents any updating by blocking CAS at this gate. Resetting the error latch permits normal operation again until the next error occurs.

Note that only the 15 high-order memory lines (A6-A20) are latched, because this defines a 64-byte block within memory, which is sufficient to locate the physical page of memory causing the error. Lisa software can map out faulty memory using the MMU in pages as small as 512 bytes.

The low-order bit in the latch is not a memory address bit. It indicates whether a CPU access or a video access was in progress when the error occurred.

For the case of a video error, only bits A15-A20 contain valid information. Bits A6-A14 are undefined.

#### 4.7.4 Processor Board Status Latch

The processor board status latch is shown at B-1 on sheet 3. It consists of an LS279 quad RS flop and an LS244 octal driver. It is interrogated by the RBES/ signal, which originates in the I/O decode logic on sheet 5.

Four of the bits are latched by the LS279. These are:

SFER/ , Soft Memory Error, which indicates a recoverable error in memory. This can be inhibited or reset by means of the SFMSK/ signal, in the processor board control register at D-2 on sheet 5.

HDER/ , Hard Memory Error, which indicates an unrecoverable error in memory. This can be inhibited or reset by means of the HDMSK/ signal from the processor board control register at D-2 on sheet 5.

VTIR/ , Vertical Retrace Interrupt, which originates in the video state machine on sheet 5. It can be inhibited by means of the VTMSK/ signal from the control register on sheet 5.

BUST/ , Bus Timeout Error, which originates in the Time Delay Logic at D-4 on sheet 3. This is reset by reading the Memory Error Address latch, which generates the RMEA/ signal.

The other three bits available are not latched and all originate in the video control logic on sheet 3. They are:

INVID, Inverted video, which can be hand-wired to provide the inverse of the video data normally presented to the video board.

VID, Video Data, which is a direct sense of the current bit stream being presented as data to the video board.

CSYNC/ , Composite Synchronization pulse, which originates in the video state machine as a composite of horizontal and vertical retrace signals.

#### 4.7.5 Time Delay Logic

The time delay logic is shown at D-4 on sheet 3 and serves two purposes:

- \* Generation of Power-on Reset
- \* Detection of Bus Time-out.

The logic consists of a 556 dual timer, with some associated circuitry.

The Power-on Reset (POR) signal is generated whenever the Reset Switch (RSTSW/ ) input from the system bus is pulsed low or at power-on. The POR signal is true for approximately 1 second.

The Bus Timeout (BUST/ ) signal is generated whenever a period of 30-300 microseconds elapses with the Address Strobe (AS/ ) signal occurring. This usually indicates that the CPU is awaiting a response from a peripheral which is either not present or is unable to respond.

## CHAPTER 5. MEMORY BOARDS

---

The Lisa provides a flexible memory configuration. The physical memory space is divided into three spaces: main memory, I/O space, and special I/O space.

The main memory occupies 2 Mbytes of space and is further subdivided into two spaces of 1 Mbyte each, which correspond to each of the two memory boards that may be present in the Lisa. Each memory board is capable of sensing in which of the two slots it is presently located because one pin of the board socket is grounded on slot 2 and left open on slot 1. The lower memory board occupies slot 2 and the higher occupies slot 1.

In order to have memory that is contiguous, when each board can not necessarily have a full 1 Mbyte available on it, a scheme is implemented whereby both boards begin at their mutual boundary of 1000000@16 and fill outward from this point to the capacity of the board.

To accommodate partially-stuffed boards and to maintain interchangeability between slots, the physical top row of the board in slot 2 (the lower board) must be the top of that board's address space. Conversely, the top row of the board in slot 1 (the upper board) must be the bottom of that board's address space.

The amount of memory in slot 2 determines the physical starting address of the memory available in the Lisa. Likewise, the amount of memory in slot 1 defines the physical ending address. Since the current memory boards are designed around 64 Kbit dynamic RAMs, the smallest possible increments are of 128 Kbytes each, or 64 Kwords of 16 bits each.

Because both the beginning and ending physical addresses are a function of the amount of memory present, the boot ROM must contain a routine that establishes the size and the location of the memory available in each particular Lisa. The routine must then configure the MMU RAM to place some part of the physical address at logical address zero.

Since the memory boards can be installed in any order or configuration, the operating system must add the required base address when modifying the MMU RAM contents for any reason.

Note that the Video Page Address latch contains a physical address, which must take the actual physical memory configuration into account.

### 5.1 Memory Block Diagram

Since each memory board is functionally identical, this section discusses the block diagram of a single board only. A memory board consists of:

- \* An address decode section
- \* The memory matrix itself, and
- \* A parity detection and generation section.

The bulk of the memory control and timing signals are generated on the processor board and provided on the memory bus. Refer to section 4.6 for a discussion of how these signals are generated. Figure 5-1 shows the memory block diagram.

Figure 5-1. Memory Block Diagram

The matrix itself consists of four rows of eighteen 64 K by 1 bit dynamic RAM devices. The extra 2 bits are used as parity bits to check the integrity of data being read from the memory.

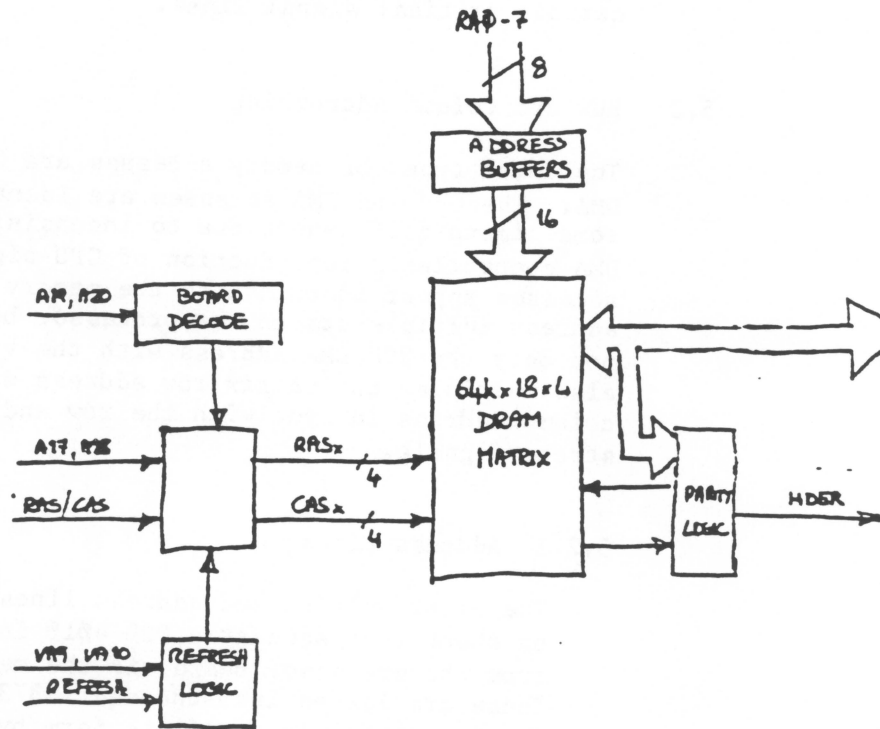


Figure 5-1. Memory Block Diagram



The parity logic both generates and checks the parity of data being written and read respectively. The decode logic is used to interpret the control signals to allow the correct RAS and CAS signals to select the proper row.

The latches and drivers are used to reduce loading on certain critical signal lines.

## 5.2 Row and Column Addressing

The three types of memory accesses are CPU, video, and DMA. The CPU and DMA accesses are identical, except for some timing differences due to inconsistencies in the DMA controller's reproduction of CPU signals. All accesses appear identical to the memory board. The address multiplexers on the processor board multiplex not only the CPU/DMA address with the video address, but also multiplex the matrix row address with the matrix column address in sync with the row and column address strobe signals.

### 5.2.1 Address Lines

The eight multiplexed address lines, RA1-RA8 at D-4 on sheet 1 of schematic 050-4010 in Appendix B, come from the processor board via the memory bus on P1. These are loaded into the dual S373 transparent latches at D-3 in duplicate form by the LTCH/ signal.

The LTCH/ signal originates in the JK flop at C-3, and is the DOTCK signal divided by two. It is active only when CAS/ is deasserted to ensure that the address for the RAMs is stable for a sufficient period.

The address lines to the matrix consist of the duplicated and latched RA1-RA8 signals, organized as upper and lower address lines to minimize signal loading and delay. These are shown at D-2 on sheet 1.

### 5.2.2 Slot Decode

The SLOT signal is shown at C-4 on sheet 1. Depending on the slot in which the board is located, this is pulled to GROUND in slot 2 (low-order board) or allowed to be pulled up to +5 V in slot 1 (high-order board).

SLOT is presented to the matrix decode logic and also to the pair of LS02 gates at C-3. The other leg of these gates is a decode of the A19 and A20 terms by the LS139 decoder at B-3.

As can be seen from the connection of the Y1 and Y2 outputs to the gates, a logic high will be output from the S32 gate at C-3 if either:

- \* SLOT and A20 are low while A19 is high
- \* SLOT and A20 are high while A19 is low.

This selects between the two boards using A19 and A20. The addressing system is shown schematically in Figure 5-2.

board	signal	row		
B1	CAS3	E	128 Kbytes	Upper 1M
B1	CAS2	D	128 Kbytes	fills
B1	CAS1	C	128 Kbytes	(SLOT 1)
B1	CAS0	B	128 Kbytes	upward
B2	CAS0	B	128 Kbytes	Lower 1M
B2	CAS1	C	128 Kbytes	fills
B2	CAS2	D	128 Kbytes	(SLOT 2)
B2	CAS3	E	128 Kbytes	downward

Figure 5-2. Memory Address Decoding

Note that the output signal from this section is used to gate only the CAS signal to the matrix. This is because no change is made to the contents of RAM when a RAS-only cycle is performed.

### 5.2.3 Matrix Device Decode

The A17 and A18 signals from the memory bus are presented to the S138 3-to-8 decoder at B-3. The third input is the SLOT signal from the previous subsection. This results in two groups of four outputs, the high-order four for the high-order board slot and the low-order four for the low-order board slot.

These signals are presented to the two rows of gates that follow, to allow memory to be physically located next to the 1000000@16 boundary discussed in the introduction to this chapter.

Note that the RAS0/ output to the matrix on sheet 1 is a product of the Y4 decode output if SLOT is high, but a product of the Y3 output if SLOT is zero. That means that it corresponds to the high-order block of the matrix in the low-order slot, but the low-order block in the high-order slot. Examination of all the other matrix strobe signals will reveal a similar pattern.

#### 5.2.4 Matrix Address Strobes

The RAS/ signal from the processor board enters the memory board from the memory bus at A-4. It is used to gate whichever of the inputs to the four S00s at A-2 has been enabled by the matrix decode logic, described in the preceding subsection.

$(\text{SLOT}/!\text{.A20}/!\text{.A19}) \rightarrow \text{SLOT } 2$

$(\text{SLOT}!\text{.A20}!\text{.A19}/) \rightarrow \text{SLOT } 1 \text{ (high order)}$

$(\text{A17}/!\text{.A18}/!\text{.CAS}/!\text{.SLOT}) + (\text{A17}!\text{.A18}!\text{.CAS}/!\text{.SLOT}/) \rightarrow \text{CAS0}/$

$(\text{A17}!\text{.A18}/!\text{.CAS}/!\text{.SLOT}) + (\text{A17}/!\text{.A18}!\text{.CAS}/!\text{.SLOT}/) \rightarrow \text{CAS1}/$

$(\text{A17}/!\text{.A18}!\text{.CAS}/!\text{.SLOT}) + (\text{A17}!\text{.A18}/!\text{.CAS}/!\text{.SLOT}/) \rightarrow \text{CAS2}/$

$(\text{A17}!\text{.A18}!\text{.CAS}/!\text{.SLOT}) + (\text{A17}/!\text{.A18}/!\text{.CAS}/!\text{.SLOT}/) \rightarrow \text{CAS3}/$

This scheme permits the memory space to be filled from the 1 Mbyte midpoint in both directions. This eliminates gaps in the physical address space when the full 2 Mbytes of memory are not installed.

The third input leg is to the S10 gates at A-3 which drives the S00's derives from the decode of VA9 and VA10 from the LS139 2-to-4 decoder at A-3. This is enabled by the Refresh (RFSH/ ) signal. Since the video address register increments sequentially, it is used to provide refresh.

The CAS/ signal from the processor board enters the memory board at C-4 on sheet 1 and gates the

selected CASx signal, provided that the LS00 gates at B-3 are enabled.

Only one CASx/ will be active during any one access. During a CPU or DMA access, only one RASx/ per board will be active, but in a video access, a second RASx/ will be active to perform matrix refresh.

### 5.3 Data and Parity

The logic concerned with data transfer and parity is located on sheet 2 of schematic 050-4010. Parity is stored in odd form for each byte in the memory matrix and generates a hard memory error if read as an even parity. Currently, there is no error correction circuitry implemented in the Lisa memory. As a consequence, soft memory errors are not corrected by the memory board.

#### 5.3.1 Memory Data Lines

The bidirectional data lines on the memory bus MD0-MD15 are connected directly to the DxIN inputs of the matrix.

They are also connected to two LS280 parity generator/checkers at C-3 and A-3, which generate the odd parity for each of the upper and lower bytes being written into memory. The EVEN outputs are written into the corresponding bit in the matrix, resulting in an 18-bit word being used to store the two bytes, each with odd parity.

Data read from the matrix appear at the DxOUT outputs, and are passed into the two LS373 transparent octal latches shown at A-2 and C-2. The data are latched on the trailing edge of the main RAS signal.

At the same time, the corresponding two parity bits are latched into the LS375 shown at C-4 and A-4. These are then gated with the MREAD signal to ensure that the ninth bit is in fact stable during a write operation before being presented to the parity generator/checkers to establish if parity is good for both bytes being read.

Output data are enabled onto the MD0-MD15 lines of the data bus by the S03 AND gate output at C-3. This term is asserted if both MREAD and LBDSL are asserted. The LBDSL originates at C-3 on sheet 1.

It shows that the board has been selected clocked on the trailing edge of RAS.

A single byte can be written to or read from memory under control of the LDS/ and UDS/ bus signals at C-4 and A-4 respectively. The other half of the memory word is read from memory, but is not passed through the LS373.

### 5.3.2 Memory Parity

Odd, byte-wide parity is generated and tested in the Lisa memory by means of two LS280 devices. Each data byte has its own device associated with it. The lower byte generates the PIL bit at C-2, while the upper byte generates the PIU at A-2.

When a word is read from memory, the corresponding two parity bytes are read and latched while the parity is checked. For a word access, if either byte results in an even parity, the JK flop at B-3 is set. This results in a HDER/ signal being presented to the processor board to denote a memory error. For a byte access, parity is checked only for the byte being accessed. The parity of the other byte is ignored.

The HDER/ signal is also fed back into the parity logic for use in parity diagnostics. Note that it is also latched until a write operation is performed to clear this.

If a parity error occurs during video access to memory, it is reported to the CPU 60 times per second until the parity is remasked or the word causing the error is rewritten. This is true even if the CPU itself never accesses the memory location causing the parity error.

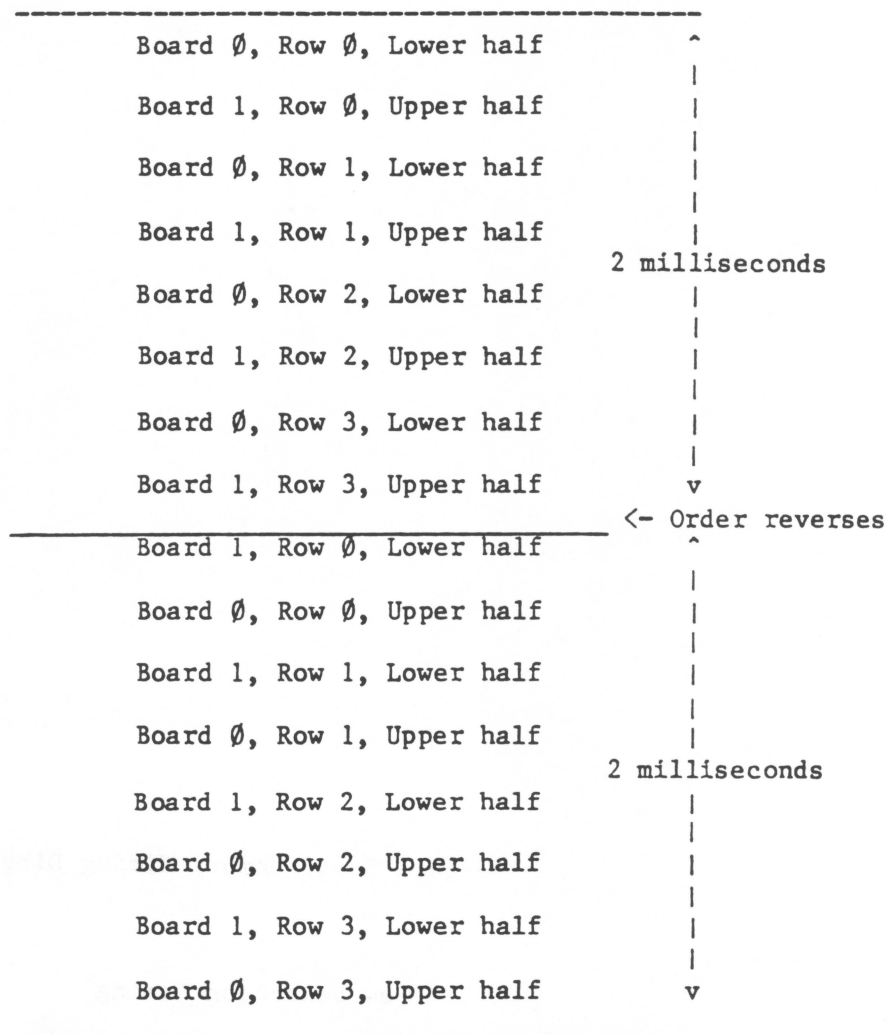
### 5.3.3 Memory Refresh

Refresh cycles occur during a video access by selecting multiple rows of memory devices on the board. Since the video addresses cycle through all device-row addresses every 128 accesses, this feature is used in place of a separate hardware refresh address generator.

The upper and lower halves of 64 Kbit RAM devices are selected by the polarity of the RA8 input at RAS time. RAM devices that require refresh every 128 cycles have their upper and lower halves refreshed

simultaneously and therefore require only 2 milliseconds for full refresh. RAM devices that require refresh every 256 cycles need the full 4 milliseconds refresh cycle time for full refresh.

Refresh alternates between installed memory boards every 128 cycles. The order of alternation is reversed after every refresh pass in order to supply the correct refresh to 256-cycle refresh RAMs, should these be installed. This is done by means of the VA8 and VA11 signals from the processor board and results in the pattern shown in Figure 5-3.



Pattern repeats from the top

Figure 5-3. Memory Refresh Pattern

#### 5.4 Memory Timing

As with most systems, the timing associated with memory access is one of the critical constraints around which the computer has been designed. The timing diagram for the memory board is shown in Figure 5-4.

Figure 5-4. Memory Timing Diagram

##### 5.4.1 Row Selection Timing

The most critical timing in the memory involves the decode of the A17 and A18 address lines. Both these lines are the product of the address translation within the MMU on the processor board, and consequently reach the memory board with some delay



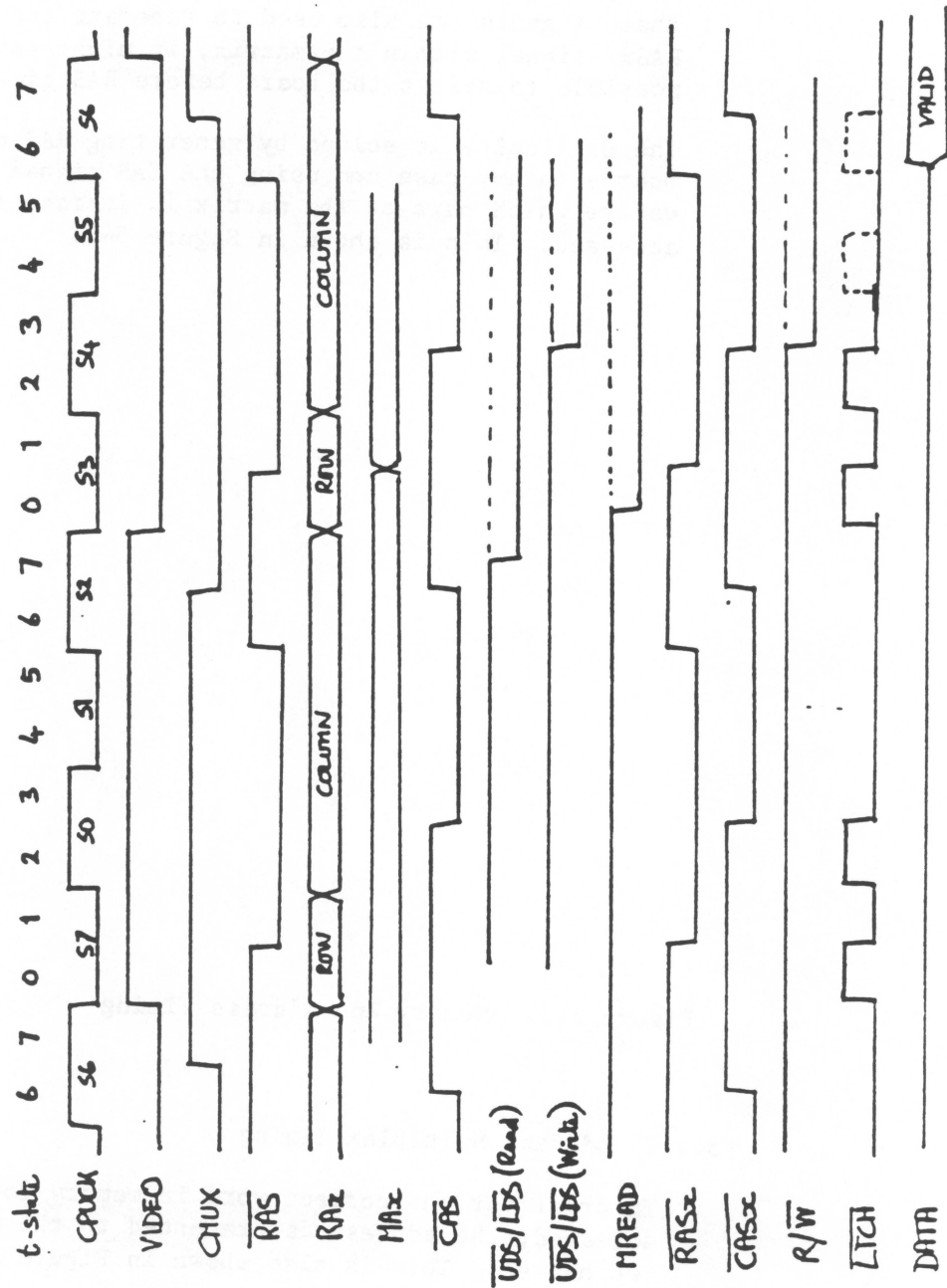


Figure 5-4. LISA Memory Timing Diagram

after the AS/ signal that begins the cycle. Since these signals are also used to generate the correct RASx/ signal within the matrix, it might not be possible to select the board before RAS time.

The difficulty is solved by generating RAS on both boards in any case and using the CAS signal to define which part of the matrix is in fact being accessed. This is shown in Figure 5-5.

Figure 5-5. Memory Row Address Timing

#### 5.4.2 Address Multiplex Timing

In order for the correct word in memory to be accessed, the address is presented to the matrix in two halves. This is also shown in Figure 5-5.

The two halves correspond to the row address and the column address of the word being accessed. The processor board makes each of these available in turn on the RA1-RA8 lines. It uses the RAS/ or CAS/ signals to strobe the address.

The multiplexed address is latched into the S373s with the LTCH/ term. This has the waveform shown in Figures 5-4 and 5-6. The flop generating it toggles at the frequency of DOTCK whenever the CAS/ term is deasserted. Otherwise, it remains asserted.

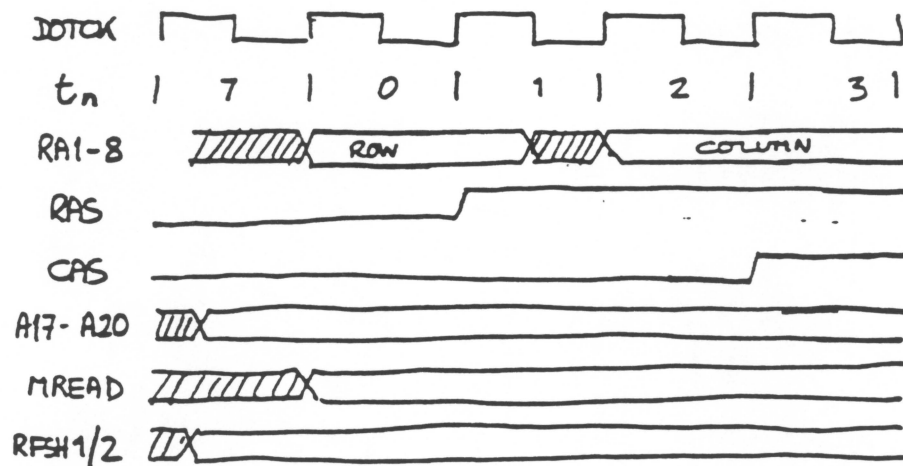


Figure 5-5. Memory Row Address Timing

#### 5.4.3 Data and Parity Timing

The constraints on the data and parity signals of the matrix are less rigorous than those for the address selection. The timing is shown in Figure 5-6.

Figure 5-6. Memory Data and Parity Timing

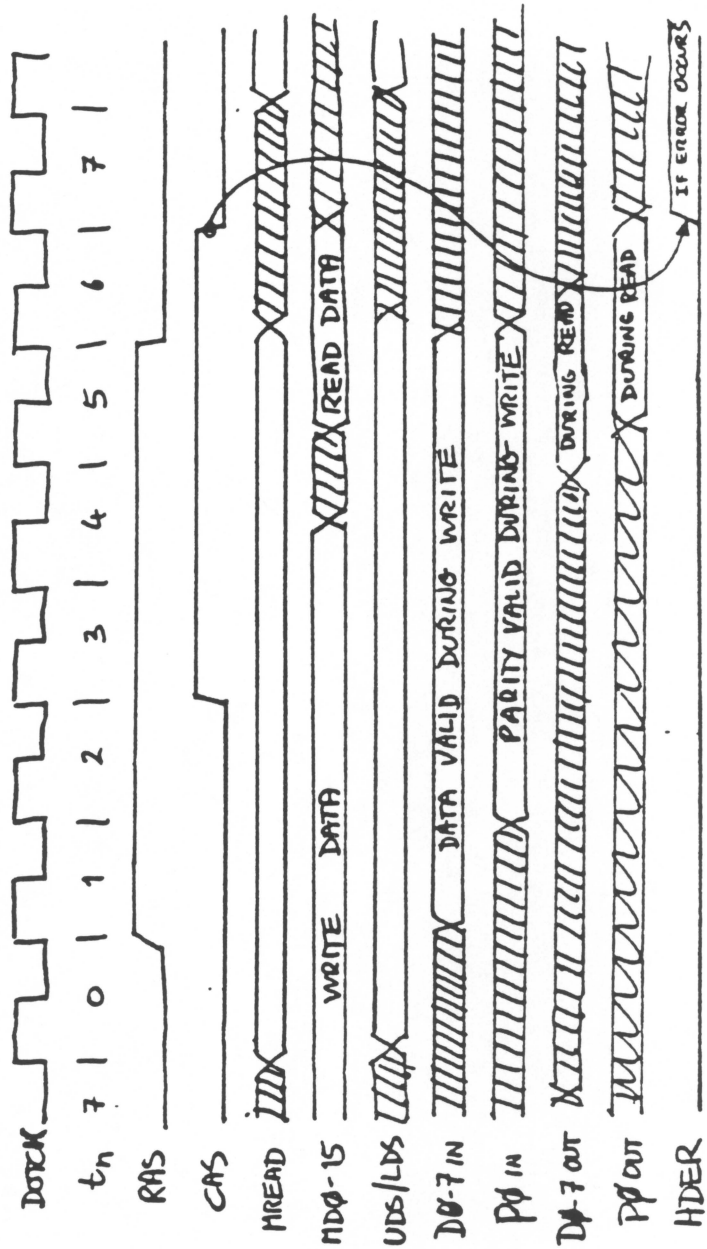


Figure 5-6. Memory Data and Parity Timing



## CHAPTER 6. THE I/O BOARD

---

The Lisa communicates with its peripherals principally through the I/O board, which contains the peripheral controllers for the external interface connectors of the Lisa. The I/O board must be present in any Lisa.

### 6.1 I/O Board Block Diagram

The main features on the I/O board are shown in Figure 6-1 and can be itemized as follows:

- \* Floppy-disk controller
- \* Two serial I/O port interfaces
- \* A parallel port interface
- \* Keyboard/Mouse interface
- \* Miscellaneous logic.

The data bus on the I/O board consists of three separate sections, which are also shown in Figure 6-1. These are the:

- \* System-bus data lines from off-board,
- \* Disk bus, internal to the floppy-disk controller,
- \* D-bus connecting to all other peripherals.

Note that some control functions are grouped together on the same interface device. This is done both for convenience and to minimize hardware.





Figure 6-1. I/O Board Block Diagram

## 6.2 Floppy-Disk Controller

The floppy-disk controller is designed around a 6504A processor and its associated memory and logic. Figure 6-2 shows a block diagram of the floppy-disk controller. Operation of the controller is described in section 6.3; The logic schematic can be found on sheet 4 of schematic 050-4008 in Appendix C.

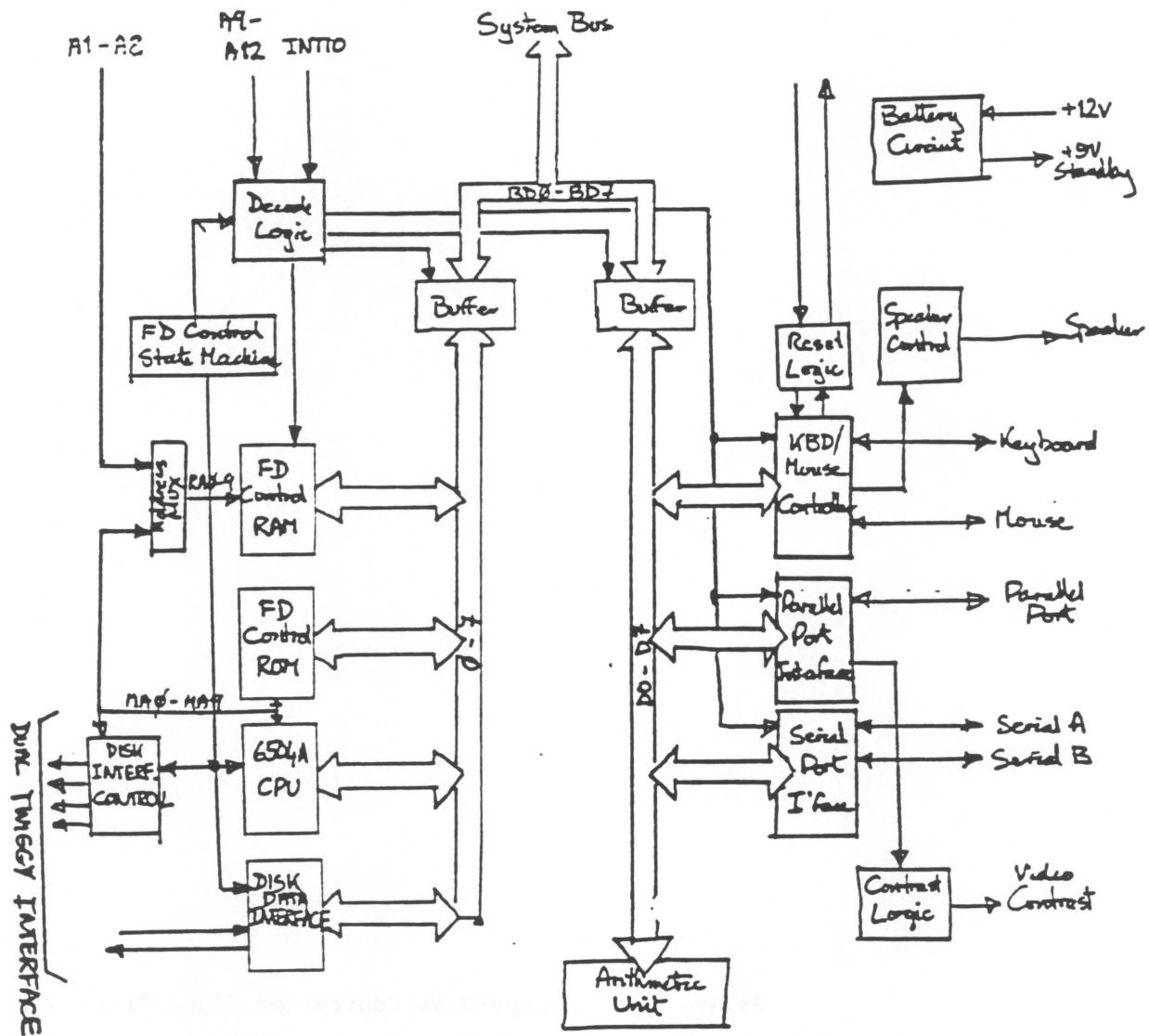


Figure 6-1. I/O Board Block Diagram

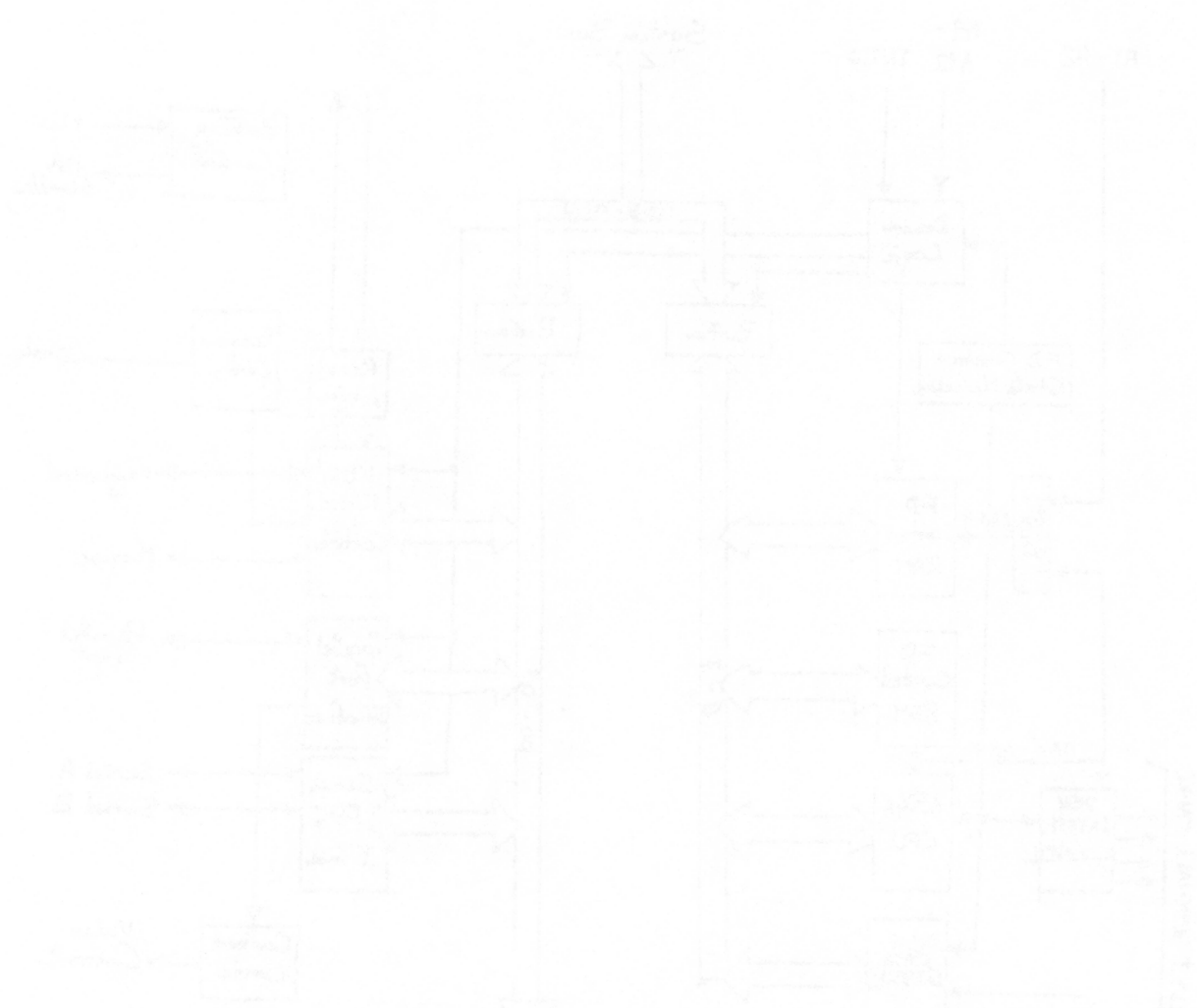


Figure 6-2. Floppy-Disk Controller Block Diagram

The controller communicates with the Lisa through an area in its dedicated memory. The CPU places commands and data to be written on the disk at locations in this RAM. The CPU can also access status information and data read from the disk that is placed in the RAM by the processor.

Timing for the controller is generated asynchronously to the 68000 by a clock generator internal to the controller. This provides clocking to the processor and to control latches.

The control lines to the two floppy-disk drives are generated by addressable latches that are loaded by

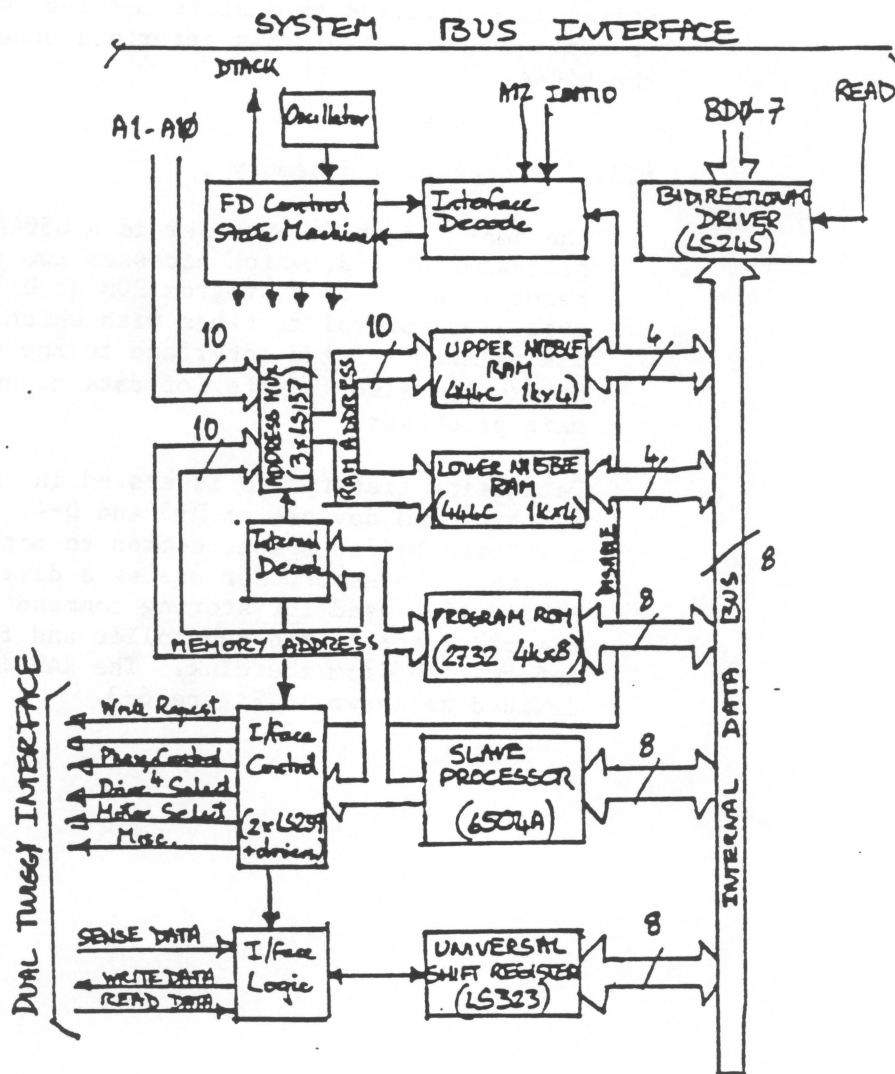


Figure 6-2. Floppy Disk Controller Block Diagram

slave processor operation. The data interface to the drives is controlled by a state machine, which is used to monitor and control this interface under control of the 6504A.

#### 6.2.1 Processor and Memory

The heart of the controller is a 6504A 8-bit processor at D-3, which accesses two types of memory. A 4 K by 8 program ROM at D-3 contains the processor control routines with which it performs manipulation of the interface to the two floppy-disk drives and also transfer of data to and from the main processor.

Data being transferred is stored in the two parallel 1 K by 4 RAM devices at D-3 and D-4. These provide a 1 Kbyte buffer space, common to both the 6504A and the main processor, for use as a data buffer. The RAM is also used for storing command strings from the processor to the controller and for status information from the disk. The RAM is functionally divided as shown in Figure 6-3.



Figure 6-3. Floppy-Disk Controller Address Space

The bidirectional LS245 octal buffer at D-4 divides the internal floppy-disk controller data bus from the system bus data lines BD0-BD7. The internal bus connects to the 2732 PROM, the 6504A processor, and the upper and lower nibbles are connected to one 444C RAM each. In addition, the bus connects to the LS323 universal shift register at D-2.

The 8 Kbyte address space of the processor is addressed by means of the internal address bus MA0-MA12. These connect directly to the program ROM and are multiplexed with system bus address lines for accesses to the buffer RAM.

0000	I/OB (Input/Output Control Block) (see Figure 6-8)
0010	R/W Shared RAM (Initialized by the 6504)
0020	6504 Status Block (Read-only for the 68000)
0030	Internal I/OB (Used only by the 6504)
0040	6504 Internal Variables (Global and Local)
00C0	LISA Parameter Storage (Used only by the 68000)
0100	6504 Stack (Used in this order) ↑
0180	6504 General Storage Area
01F4	I/O Buffer
05FF	(524 bytes)

Figure 6-3. Floppy Disk Controller Address Space



The internal memory bus is also used to select the control lines to be manipulated on the disk interface via the LS259 selectable latches at B-1 and C-2. It provides internal decodes using the LS139 dual 2-to-4 decoder shown at B-2 and B-3.

### 6.2.2 System Bus Interface

The interface to the system bus consists of the devices shown on the left side of sheet 4 of the schematic, that is, the bidirectional data bus transceiver at D-4, the three LS157 2-to-1 Multiplexer devices at C-4 to A-4, and some control circuitry at A-3 and A-4.

The processor is capable of locking out any communication on the main bus by means of the DIS signal. This is done because many routines within the disk controller are time-critical. Accesses from the 68000 would render them impossible.

DIS is one of the inputs to the LS32 gates at A-3. When this signal is asserted, the J input of the JK flop at A-3 is always asserted. This means that the output is always true, and that the second JK flop at A-3 is always set, giving an output at pin 6 which is always asserted. This forces the S input on the LS157 Multiplexer to select the MAX inputs at all times and disables the bus transceiver at D-4.

The Q/ output of the second LS109 flop at A-3 is gated into another LS109 JK flop at A-2. This generates the DTACK/ signal on the system bus only if the Q/ output from A-3 is high. This can happen only when DIS is deasserted.

Thus any attempt to communicate with the controller while DIS is high does not result in a DTACK/ acknowledgement signal. The access cycle hangs, waiting for this signal until the bus timeout on the processor board triggers and the BERR/ signal is generated. Refer to section 4.7 for a discussion of this.

Note that the DTACK/ signal is generated by an LS367 tristate driver device, which is enabled only when the Flop at A-3 Q output is low.

### 6.2.3 Timing Generation

The floppy-disk controller runs asynchronously to the Lisa. Its timing is controlled by logic built

around an LS161A 4-bit counter.

Timing is provided by a 16 MHz oscillator, located at B-3 on sheet 4 of schematic 050-4008. The output is buffered by the Schmitt trigger NAND gates before being presented to the LS161A counter at B-3. Pin 5 of device U6A may be used to provide an alternative clock input if pin 1 is pulled to ground to disable the normal clock.

The timing state sequence that the counter passes through is dependent on whether the controller is communicating with the processor or performing internal control functions with the processor locked out. The state machine goes through eight states in the former case and nine in the latter. Refer to Figure 6-4 for an overview of the states that are passed through.

Figure 6-4. Floppy-Disk Counter Timing States

If the controller is communicating with the processor, the DIS signal is deasserted. This means that the LS109 JK flop at A-3 is in the reset condition, resulting in a low condition on the P0 input to the LS161A at B-2.

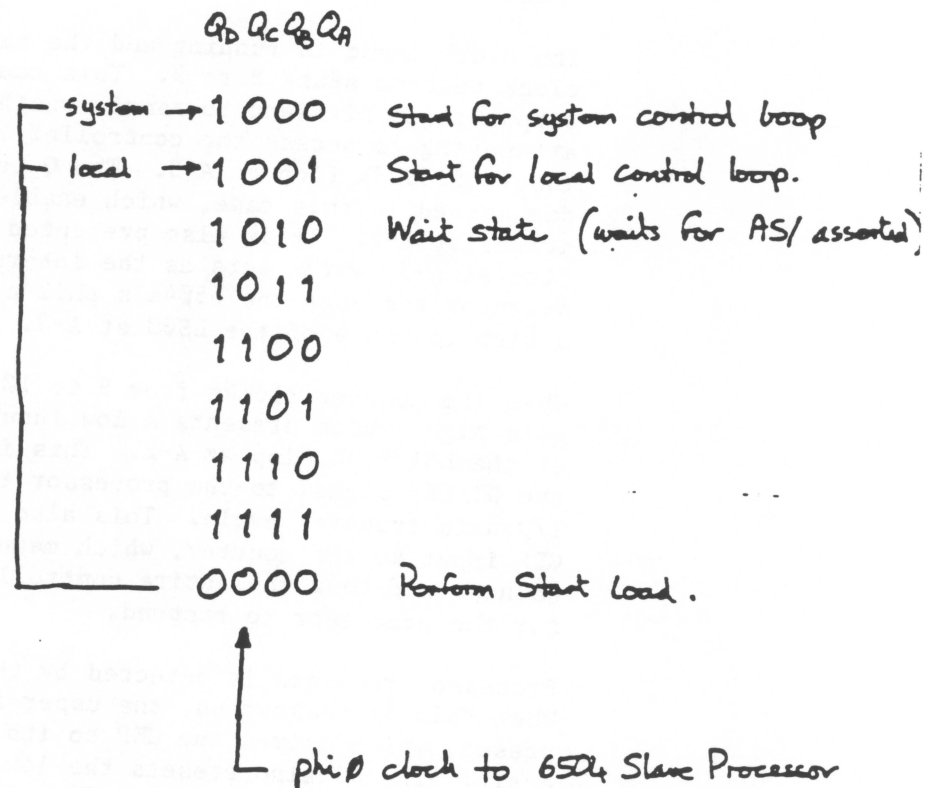


Figure 6-4. FD Counter Timing States

When the counter rolls over to zero, the Q3 output goes low and causes the P-inputs to be loaded. In this case, an 8 is loaded, since the P3 input is pulled high and all other inputs are low. At this point, the Q1 output is low, which results in the CEP input being asserted. Since the CET input is pulled high, the counter is enabled and begins to count.

The clock input is running and the machine will clock up from state 8 to 9. This causes the Q0 output to go high and to sample whether the CPU is attempting to access the controller by clocking the lower LS109 JK flop at A-3. The Q output becomes deasserted in this case, which enables the DTACK driver at A-2. It is also presented to the upper flop at A-3, which acts as the internal mode selector and uses the 6504A's phi2 clock to present a high to pin 4 of the LS00 at A-2.

When the counter clocks from 9 to 10, the Q1 output goes high, which presents a low input to the j input of the LS109 JK flop at A-2. This in turn asserts the DTACK/ signal to the processor to initiate the I/O data transfer cycle. This also deasserts the CEP input to the counter, which means that the counter and thus the entire controller hangs waiting for the processor to respond.

Processor response is detected by the AS signal. When this is deasserted, the upper flop at A-3 is preset, which drives the CEP to its asserted state again. The AS also presets the lower flop, which disables the DTACK/ signal. The counter then proceeds in sequence through states 11 to 15 before rolling over and beginning the sequence again.

In the case where the processor is not attempting to access the controller, the lower flop at A-3 is set. This causes the counter to begin in state 9, since the P0 input is high when the PE load signal is asserted.

The Q0 output is solely used to clock the disable flop at A-3. Due to the asynchronous nature of this clocking with the interface, the second flop at B-3 has been added to give a stable period to the state.

The Q1 output performs two functions. First, it is used as a clock to drive the disk state machine at D-1 and D-2. Second, it is used as a gating term for the DTACK flop at A-2, which also halts the counter to wait for processor response.

The Q2 output is only used to provide the input clock to the processor at C-2. This defines the period of the phi2 clock output by the processor.

The Q3 output is used exclusively to reset the counter to its initial count state.

The phi2 clock from the 6502A is used to clock the upper flop at A-2 and also to provide timing for loading the control latches at B-1 and C-2.

#### 6.2.4 Disk State Machine

The interface to the drive itself includes a state machine whose purpose it is to perform the data parallel/serial translation during transfer to and from the drive.

The state machine consists of three devices, as shown in Figure 6-5. The central device is a 256-by 8-bit ROM at D-2, which works in conjunction with the hex D-Flop LSI74 at D-1 and the 74LS323 8-bit universal shift register at D-2.

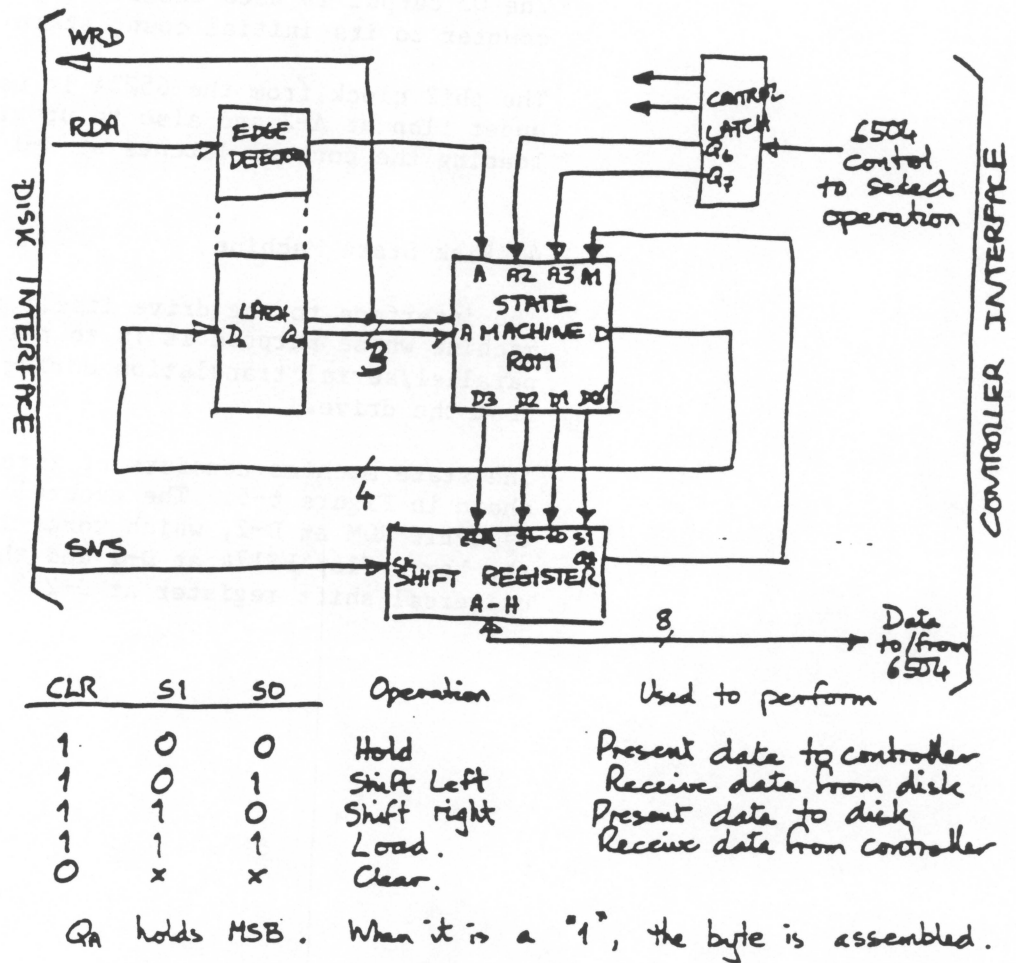


Figure 6-5. Disk State Machine

~~Figure 6-5. Disk State Machine~~

The processor uses two outputs of the LS259 to select one of four operations within the state machine. In addition, data are made available on the internal controller data bus for parallel loading into the shift register.

The state machine is clocked by the Q2 output of the timing counter described in subsection 6.2.3 above.

The interface signals that connect to this section of the logic are as follows:

RDA is input from the drive and is the Read Data line containing serial data from the selected head and drive.

WRD is output to the drive and is the Write Data line for the serial data written by the selected drive and head.

SNS is input from the drive and is the sense line that presents the selected status information on the floppy-disk drive system.

#### 6.2.5 Stepper Motor Control

The control of the stepper motor in the drives is performed with the LS259 8-bit addressable latch at C-2. The various bits are selected by means of the low-order internal controller address lines.

These are passed through line drivers to provide the following signals. Note that the first two signals do not actually control stepper movement. The signals are:

WRQ/ is the Write Request line, which is output to the drive to permit write operations to be performed.

HDS is the Head Select line, which is output to the drive to indicate which of the two heads on the selected drive is to be used to write or read data. It also is used to select which status data are to be read on the sense line.

phi0-phi3 are the four phase control lines output to the drive to control the carriage



motion performed by the stepper motor. In addition, the  $\phi$  signal is used in the selection of status data read on the sense line.

The first two signals are placed here only for convenience. The motor phase control line operation is described in subsection 6.3.5

#### 6.2.6 General Drive Control

The function of the general drive control logic is to manipulate lines on the drive interface. This is performed by the LS259 8-bit addressable latch at B-1 with some additional logic.

The latched bits are addressed by the low-order four lines on the internal controller address bus.

Two bits are used to provide information to the CPU:

FDIR is the Floppy-Disk Interrupt Request. It is fed into the inverting driver shown at B-4 on sheet 3 where it generates the IOIR/ interrupt request to the processor. It is also presented to the PB4 inputs of the keyboard interface device for polling by the CPU.

DSKDIAG is the Disk Diagnostic line that is presented to the PB7 input of the parallel port interface device for polling by the CPU to indicate that the controller is performing disk diagnostics.

One bit is used for internal control on the I/O board:

Disable (DIS) is the control signal that the 6504A uses to inhibit the interface to the 68000 when it is the process of an operation which must not be interrupted.

Four bits are used as control signals to the disk drives:

MT0/ & MT1/ are the motor control lines, which output to the two individual disk drives to control the spindle rotation

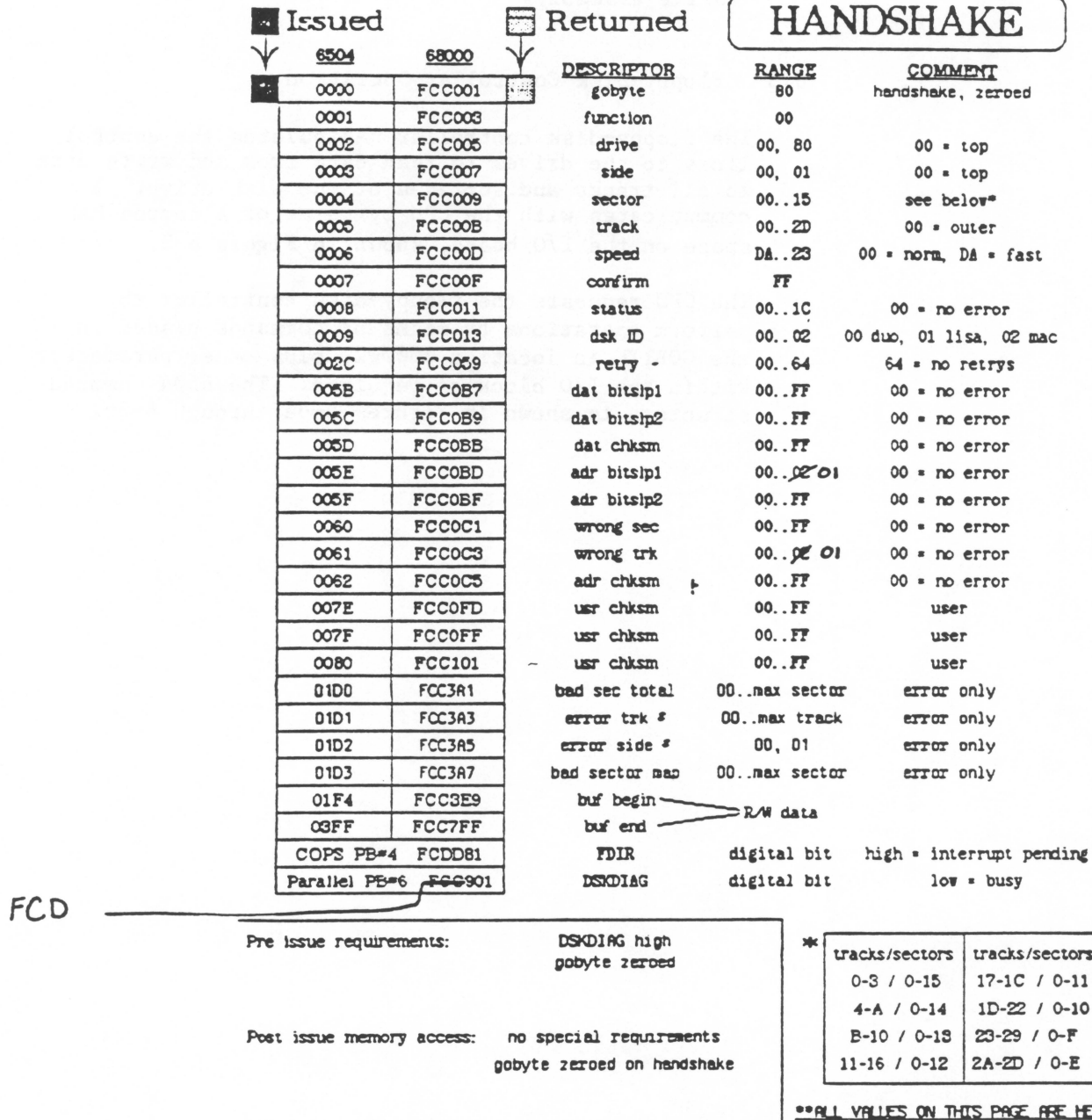
speed.

DR0/ & DR1/ are the drive select lines, which select between the two drives in the dual drive assembly.

### 6.3 Floppy-Disk Controller Operation

The floppy-disk controller manipulates the control lines to the drives to read data from and write data to all tracks and surfaces of the disk drive. It communicates with the CPU by means of a common RAM space on the I/O board, shown in Figure 6-3.

The CPU requests the floppy-disk controller to perform operations by means of commands placed in the GOBYTE in location 0000@2, plus other parameters within the I/O block as required. The 6504 command structure is shown in Figures 6-6a through 6-6r.

Figure 6-6a. **FD** Controller Handshake Command

Floppy - Disk [config.]

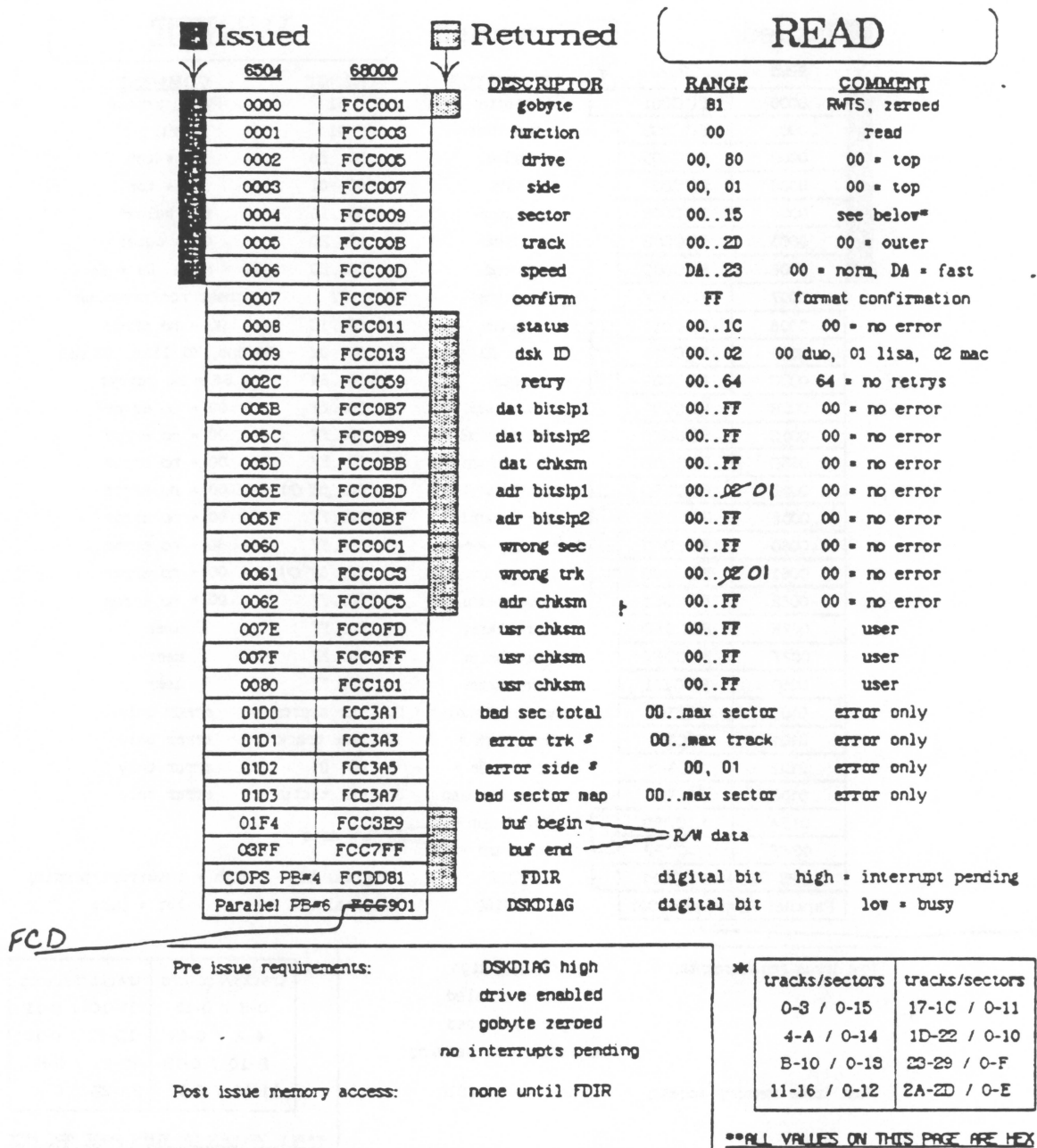


Figure 6-6b. FD Controller Read Command

Issued		Returned		WRITE	
6504	68000	DESCRIPTOR	RANGE	COMMENT	
0000	FCC001	gobyte	81	RWTS, zeroed	
0001	FCC003	function	01	write	
0002	FCC005	drive	00, 80	00 = top	
0003	FCC007	side	00, 01	00 = top	
0004	FCC009	sector	00..15	see below*	
0005	FCC00B	track	00..2D	00 = outer	
0006	FCC00D	speed	DA..23	00 = norm, DA = fast	
0007	FCC00F	confirm	FF	format confirmation	
0008	FCC011	status	00..1C	00 = no error	
0009	FCC013	dsk ID	00..02	00 duo, 01 lisa, 02 mac	
002C	FCC059	retry	00..64	64 = no retrys	
005B	FCC0B7	dat bitslp1	00..FF	00 = no error	
005C	FCC0B9	dat bitslp2	00..FF	00 = no error	
005D	FCC0BB	dat chksm	00..FF	00 = no error	
005E	FCC0BD	adr bitslp1	00..02/01	00 = no error	
005F	FCC0BF	adr bitslp2	00..FF	00 = no error	
0060	FCC0C1	wrong sec	00..FF	00 = no error	
0061	FCC0C3	wrong trk	00..02/01	00 = no error	
0062	FCC0C5	adr chksm	00..FF	00 = no error	
007E	FCC0FD	usr chksm	00..FF	user	
007F	FCC0FF	usr chksm	00..FF	user	
0080	FCC101	usr chksm	00..FF	user	
01D0	FCC3A1	bad sec total	00..max sector	error only	
01D1	FCC3A3	error trk #	00..max track	error only	
01D2	FCC3A5	error side #	00, 01	error only	
01D3	FCC3A7	bad sector map	00..max sector	error only	
01F4	FCC3E9	buf begin	R/W data		
03FF	FCC7FF	buf end			
COPS PB#4	PCDD81	FDIR	digital bit	high = interrupt pending	
Parallel PB#6	PC6901	DSKDIAG	digital bit	low = busy	

FCD

Pre issue requirements:

DSKDIAG high  
drive enabled  
gobyte zeroed  
no interrupts pending

Post issue memory access:

none until FDIR

\*

tracks/sectors	tracks/sectors
0-S / 0-15	17-1C / 0-11
4-A / 0-14	1D-22 / 0-10
B-10 / 0-13	23-29 / 0-F
11-16 / 0-12	2A-2D / 0-E

\*\*\*ALL VALUES ON THIS PAGE ARE HZ

FCD

Pre Issue requirements: DSKDIAG high  
drive enabled  
gobyte zeroed  
no interrupts pending

Post Issue memory access: none until FDIR

*		tracks/sectors	tracks/sectors
0-3	/	0-15	17-1C / 0-11
4-A	/	0-14	1D-22 / 0-10
B-10	/	0-13	23-29 / 0-F
11-16	/	0-12	2A-2D / 0-E

\*\*ALL VALUES ON THIS PAGE ARE HEX

Figure 6-6c. FD Controller Write Command

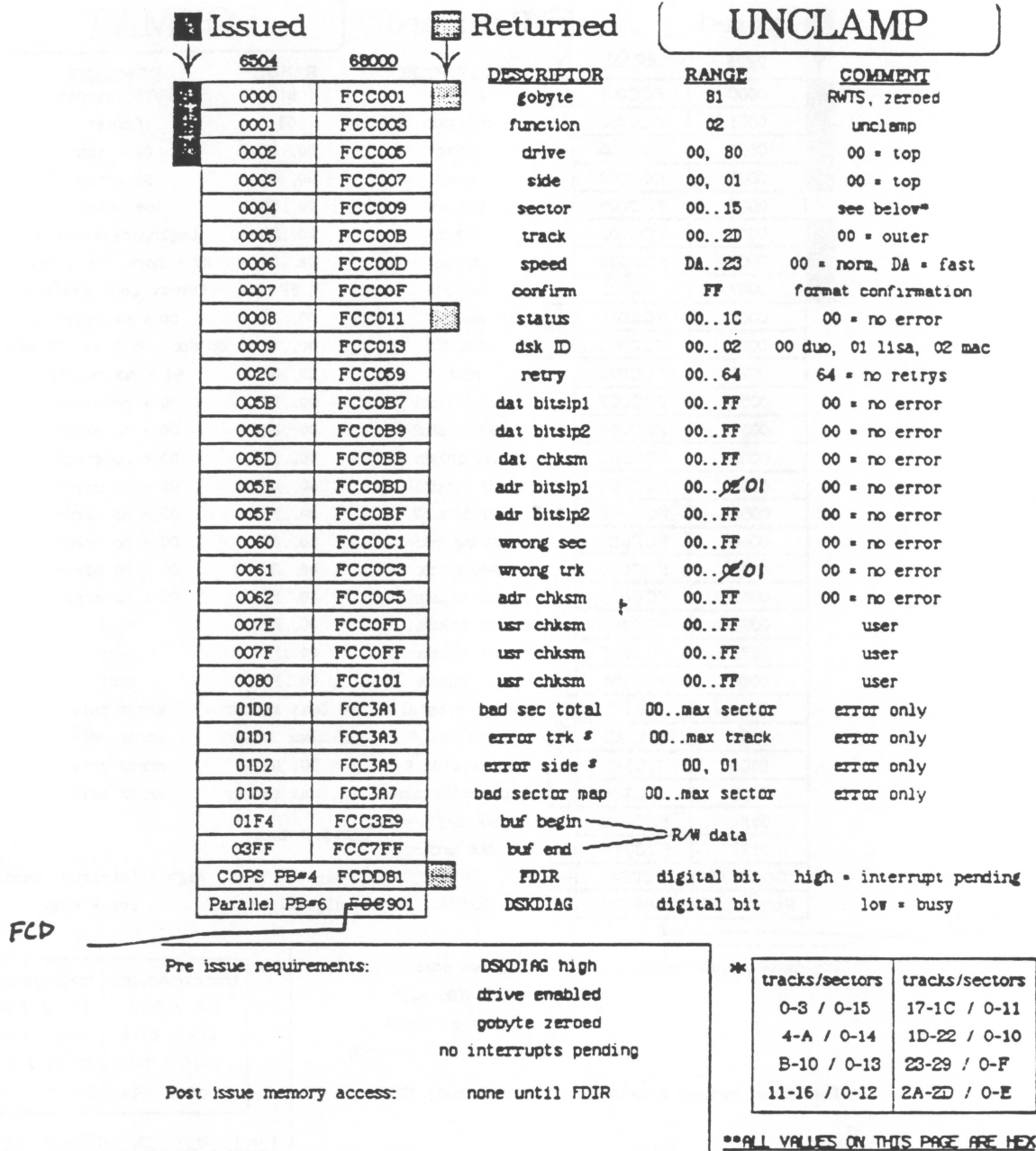


Figure 6-6d. FD Controller Unclamp Command

Issued		Returned		FORMAT	
6504	68000	DESCRIPTOR	RANGE	COMMENT	
0000	FCC001	gobyte	81	RWTS, zeroed	
0001	FCC003	function	03	format	
0002	FCC005	drive	00, 80	00 = top	
0003	FCC007	side	00, 01	00 = top	
0004	FCC009	sector	00..15	see below*	
0005	FCC00B	track	00..2D	beginning track #	
0006	FCC00D	speed	DA..23	00 = norm, DA = fast	
0007	FCC00F	confirm	FF	format confirmation	
0008	FCC011	status	00..1C	00 = no error	
0009	FCC013	dsk ID	00..02	00 duo, 01 lisa, 02 mac	
002C	FCC059	retry	00..64	64 = no retrys	
005B	FCC0B7	dat bitsip1	00..FF	00 = no error	
005C	FCC0B9	dat bitsip2	00..FF	00 = no error	
005D	FCC0BB	dat chksm	00..FF	00 = no error	
005E	FCC0BD	adr bitsip1	00..0E01	00 = no error	
005F	FCC0BF	adr bitsip2	00..FF	00 = no error	
0060	FCC0C1	wrong sec	00..FF	00 = no error	
0061	FCC0C3	wrong trk	00..0E01	00 = no error	
0062	FCC0C5	adr chksm	00..FF	00 = no error	
007E	FCC0FD	usr chksm	00..FF	user	
007F	FCC0FF	usr chksm	00..FF	user	
0080	FCC101	usr chksm	00..FF	user	
01D0	FCC3A1	bad sec total	00..max sector	error only	
01D1	FCC3A3	error trk #	00..max track	error only	
01D2	FCC3A5	error side #	00, 01	error only	
01D3	FCC3A7	bad sector map	00..max sector	error only	
01F4	FCC3E9	buf begin	R/W data		
03FF	FCC7FF	buf end			
COPS PB#4	PCDD81	FDIR	digital bit	high = interrupt pending	
Parallel PB#6	PC901	DSKDIAG	digital bit	low = busy	

FCD

Pre issue requirements:

drive enabled

DSKDIAG high

gobyte zeroed

no interrupts pending

Post issue memory access:

none until FDIR

\* tracks/sectors

0-3 / 0-15	17-1C / 0-11
4-A / 0-14	1D-22 / 0-10
B-10 / 0-13	23-29 / 0-F
11-16 / 0-12	2A-2D / 0-E

\*\*ALL VALUES ON THIS PAGE ARE HEX

FCD

Pre issue requirements:

- drive enabled
- DSKDIAG high
- gobyte zeroed
- no interrupts pending

Post issue memory access: none until FDIR

* tracks/sectors	tracks/sectors
0-3 / 0-15	17-1C / 0-11
4-A / 0-14	1D-22 / 0-10
B-10 / 0-13	23-29 / 0-F
11-16 / 0-12	2A-2D / 0-E

\*\*ALL VALUES ON THIS PAGE ARE HEX

Figure 6-6e. FD Controller Format Command



Issued		Returned		VERIFY		
6504	68000			DESCRIPTOR	RANGE	COMMENT
0000	FCC001			gobyte	81	RWTS, zeroed
0001	FCC003			function	04	verify
0002	FCC005			drive	00, 80	00 = top
0003	FCC007			side	00, 01	00 = top
0004	FCC009			sector	00..15	see below*
0005	FCC00B			track	00..2D	beginning track
0006	FCC00D			speed	DA..23	00 = norm, DA = fast
0007	FCC00F			confirm	FF	format confirmation
0008	FCC011			status	00..1C	00 = no error
0009	FCC013			dsk ID	00..02	00 duo, 01 lisa, 02 mac
002C	FCC059			retry	00..64	64 = no retrys
005B	FCC0B7			dat bitslp1	00..FF	00 = no error
005C	FCC0B9			dat bitslp2	00..FF	00 = no error
005D	FCC0BB			dat chksm	00..FF	00 = no error
005E	FCC0BD			adr bitslp1	00..01	00 = no error
005F	FCC0BF			adr bitslp2	00..FF	00 = no error
0060	FCC0C1			wrong sec	00..FF	00 = no error
0061	FCC0C3			wrong trk	00..01	00 = no error
0062	FCC0C5			adr chksm	00..FF	00 = no error
007E	FCC0FD			usr chksm	00..FF	user
007F	FCC0FF			usr chksm	00..FF	user
0080	FCC101			usr chksm	00..FF	user
01D0	FCC3A1			bad sec total	00..max sector	error only
01D1	FCC3A3			error trk #	00..max track	error only
01D2	FCC3A5			error side #	00, 01	error only
01D3	FCC3A7			bad sector map	00..max sector	error only
01F4	FCC3E9			buf begin	R/W data	
03FF	FCC7FF			buf end		
COPS PB#4	FCDD81			FDIR	digital bit	high = interrupt pending
Parallel PB#6	FC901			DSKDIAG	digital bit	low = busy

FCD

Pre issue requirements:	drive enabled
	DSKDIAG high
	gobyte zeroed
	no interrupts pending
Post issue memory access:	none until FDIR

* tracks/sectors	tracks/sectors
0-8 / 0-15	17-1C / 0-11
4-A / 0-14	1D-22 / 0-10
B-10 / 0-13	23-29 / 0-F
11-16 / 0-12	2A-2D / 0-E

\*\*ALL VALUES ON THIS PAGE ARE HEX

Figure 6-6f. FD Controller Verify Command



Issued		Returned		FORMAT TRACK	
6504	68000	DESCRIPTOR	RANGE	COMMENT	
0000	FCC001	gobyte	81	RWTS, zeroed	
0001	FCC003	function	05	format track	
0002	FCC005	drive	00, 80	00 = top	
0003	FCC007	side	00, 01	00 = top	
0004	FCC009	sector	00..15	see below*	
0005	FCC00B	track	00..2D	00 = outer	
0006	FCC00D	speed	DA..23	00 = norm, DA = fast	
0007	FCC00F	confirm	FF	format confirmation	
0008	FCC011	status	00..1C	00 = no error	
0009	FCC013	dsk ID	00..02	00 duo, 01 lisa, 02 mac	
002C	FCC059	retry	00..64	64 = no retries	
005B	FCC0B7	dat bitslp1	00..FF	00 = no error	
005C	FCC0B9	dat bitslp2	00..FF	00 = no error	
005D	FCC0BE	dat chksm	00..FF	00 = no error	
005E	FCC0BD	adr bitslp1	00..01	00 = no error	
005F	FCC0BF	adr bitslp2	00..FF	00 = no error	
0060	FCC0C1	wrong sec	00..FF	00 = no error	
0061	FCC0C3	wrong trk	00..01	00 = no error	
0062	FCC0C5	adr chksm	00..FF	00 = no error	
007E	FCC0FD	usr chksm	00..FF	user	
007F	FCC0FF	usr chksm	00..FF	user	
0080	FCC101	usr chksm	00..FF	user	
01D0	FCC3A1	bad sec total	00..max sector	error only	
01D1	FCC3A3	error trk #	00..max track	error only	
01D2	FCC3A5	error side #	00, 01	error only	
01D3	FCC3A7	bad sector map	00..max sector	error only	
01F4	FCC3E9	buf begin	R/W data		
03FF	FCC7FF	buf end			
COPS PB#4	FCDD81	FDIR	digital bit	high = interrupt pending	
Parallel PB#6	FC6901	DSKDIAG	digital bit	low = busy	

Pre Issue requirements:		drive enabled
		DSKDIAG high
		gobyte zeroed
		no interrupts pending
Post Issue memory access:		none until FDIR

* tracks/sectors	tracks/sectors
0-3 / 0-15	17-1C / 0-11
4-A / 0-14	1D-22 / 0-10
B-10 / 0-13	23-29 / 0-F
11-16 / 0-12	2A-2D / 0-E

**\*\*ALL VALUES ON THIS PAGE ARE HEX**

Figure 6-6g. FD Controller Format Track Command

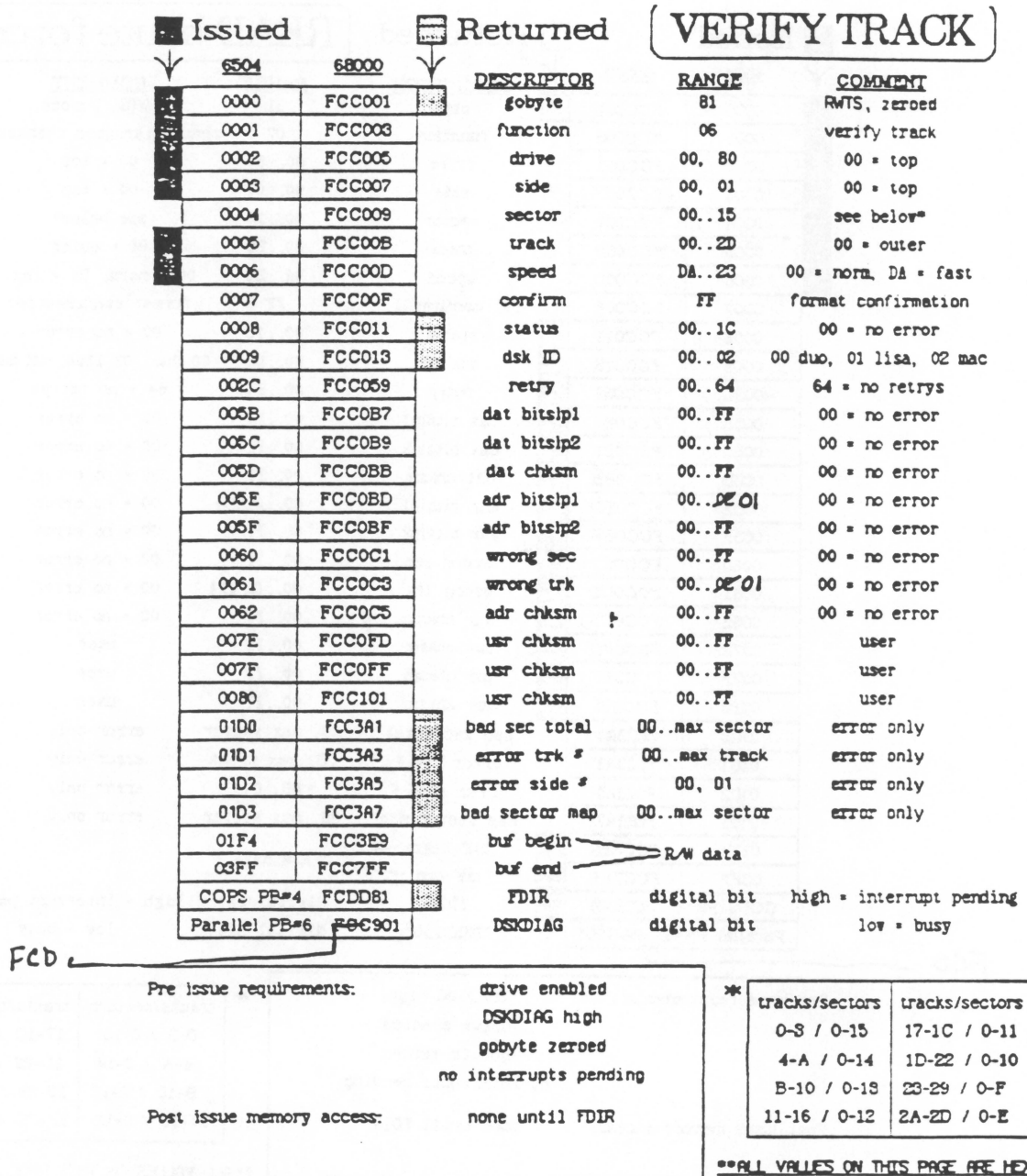


Figure 6-6h. FD Controller Verify Track Command

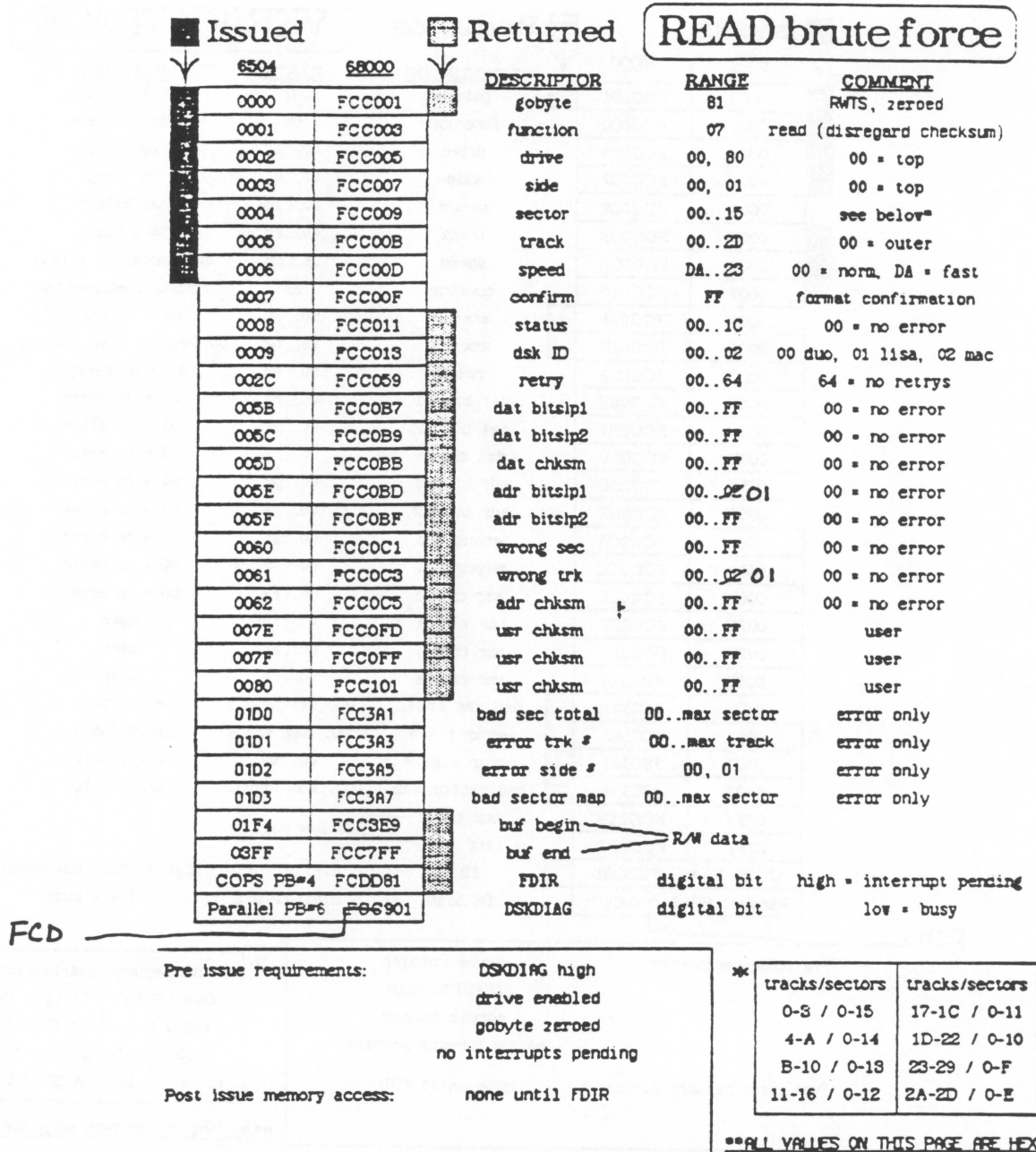


Figure 6-6i. FD Controller Read Brute Force Command

Issued		Returned		WRITE brute force	
	6504	68000	DESCRIPTOR	RANGE	COMMENT
	0000	FCC001	gobyte	81	RWTS, zeroed
	0001	FCC003	function	08	write (user checksum)
	0002	FCC005	drive	00, 80	00 = top
	0003	FCC007	side	00, 01	00 = top
	0004	FCC009	sector	00..15	see below*
	0005	FCC00B	track	00..2D	00 = outer
	0006	FCC00D	speed	DA..23	00 = norm, DA = fast
	0007	FCC00F	confirm	FF	format confirmation
	0008	FCC011	status	00..1C	00 = no error
	0009	FCC013	dsk ID	00..02	00 duo, 01 lisa, 02 mac
	002C	FCC059	retry	00..64	64 = no retrys
	005B	FCC0B7	dat bitslp1	00..FF	00 = no error
	005C	FCC0B9	dat bitslp2	00..FF	00 = no error
	005D	FCC0BB	dat chksm	00..FF	00 = no error
	005E	FCC0BD	adr bitslp1	00..01	00 = no error
	005F	FCC0BF	adr bitslp2	00..FF	00 = no error
	0060	FCC0C1	wrong sec	00..FF	00 = no error
	0061	FCC0C3	wrong trk	00..01	00 = no error
	0062	FCC0C5	adr chksm	00..FF	00 = no error
	007E	FCC0FD	usr chksm	00..FF	user
	007F	FCC0FF	usr chksm	00..FF	user
	0080	FCC101	usr chksm	00..FF	user
	01D0	FCC3A1	bad sec total	00..max sector	error only
	01D1	FCC3A3	error trk #	00..max track	error only
	01D2	FCC3A5	error side #	00, 01	error only
	01D3	FCC3A7	bad sector map	00..max sector	error only
	01F4	FCC3E9	buf begin	R/W data	
	03FF	FCC7FF	buf end		
	COPS PB#4	FCDD81	FDIR	digital bit	high = interrupt pending
	Parallel PB#6	FEE901	DSKDIAG	digital bit	low = busy

Pre issue requirements:

DSKDIAG high

drive enabled

gobyte zeroed

no interrupts pending

Post issue memory access:

none until FDIR

\*

tracks/sectors	tracks/sectors
0-3 / 0-15	17-1C / 0-11
4-A / 0-14	1D-22 / 0-10
B-10 / 0-18	23-29 / 0-F
11-16 / 0-12	2A-2D / 0-E

\*\*ALL VALUES ON THIS PAGE ARE HEX

Figure 6-6j. FD Controller Write Brute Force Command

Issued		Returned		CLAMP	
6504	68000	DESCRIPTOR	RANGE	COMMENT	
0000	FCC001	gobyte	81	RWTS, zeroed	
0001	FCC003	function	09	clamp	
0002	FCC005	drive	00, 80	00 = top	
0003	FCC007	side	00, 01	00 = top	
0004	FCC009	sector	00..15	see below*	
0005	FCC00B	track	00..2D	00 = outer	
0006	FCC00D	speed	DA..23	00 = norm, DA = fast	
0007	FCC00F	confirm	FF	format confirmation	
0008	FCC011	status	00..1C	00 = no error	
0009	FCC013	dsk ID	00..02	00 duo, 01 lisa, 02 mac	
002C	FCC059	retry	00..64	64 = no retrys	
005B	FCC0B7	dat bitsip1	00..FF	00 = no error	
005C	FCC0B9	dat bitsip2	00..FF	00 = no error	
005D	FCC0BB	dat chksm	00..FF	00 = no error	
005E	FCC0BD	adr bitsip1	00..0201	00 = no error	
005F	FCC0BF	adr bitsip2	00..FF	00 = no error	
0060	FCC0C1	wrong sec	00..FF	00 = no error	
0061	FCC0C3	wrong trk	00..0201	00 = no error	
0062	FCC0C5	adr chksm	00..FF	00 = no error	
007E	FCC0FD	usr chksm	00..FF	user	
007F	FCC0FF	usr chksm	00..FF	user	
0080	FCC101	usr chksm	00..FF	user	
01D0	FCC3A1	bad sec total	00..max sector	error only	
01D1	FCC3A3	error trk #	00..max track	error only	
01D2	FCC3A5	error side #	00, 01	error only	
01D3	FCC3A7	bad sector map	00..max sector	error only	
01F4	FCC3E9	buf begin	R/W data		
03FF	FCC7FF	buf end			
COPS PB#4	FCDD81	FDIR	digital bit	high = interrupt pending	
Parallel PB#6	FCE901	DSKDIAG	digital bit	low = busy	

Pre Issue requirements:

- DSKDIAG high
- drive enabled
- gobyte zeroed
- no interrupts pending

Post issue memory access: none until FDIR

* tracks/sectors	tracks/sectors
0-8 / 0-15	17-1C / 0-11
4-A / 0-14	1D-22 / 0-10
B-10 / 0-13	23-29 / 0-F
11-16 / 0-12	2A-2D / 0-E

\*\*ALL VALUES ON THIS PAGE ARE HEX

Figure 6-6k. FD Controller Clamp Command

Issued	Returned		SEEK
6504	68000	DESCRIPTOR	RANGE COMMENT
0000	FCC001	gobyte	83 SEEK, zeroed
0001	FCC003	function	00
0002	FCC005	drive	00, 80 00 = top
0003	FCC007	side	00, 01 00 = top
0004	FCC009	sector	00..15 see below*
0005	FCC00B	track	00..2D 00 = outer
0006	FCC00D	speed	DA..23 00 = norm, DA = fast
0007	FCC00F	confirm	FF format confirmation
0008	FCC011	status	00..1C 00 = no error
0009	FCC013	disk ID	00..02 00 duo, 01 lisa, 02 mac
002C	FCC059	retry	00..64 64 = no retrys
005B	FCC0B7	dat bitslp1	00..FF 00 = no error
005C	FCC0B9	dat bitslp2	00..FF 00 = no error
005D	FCC0BB	dat chksm	00..FF 00 = no error
005E	FCC0BD	adr bitslp1	00..01 00 = no error
005F	FCC0BF	adr bitslp2	00..FF 00 = no error
0060	FCC0C1	wrong sec	00..FF 00 = no error
0061	FCC0C3	wrong trk	00..01 00 = no error
0062	FCC0C5	adr chksm	00..FF 00 = no error
007E	FCC0FD	usr chksm	00..FF user
007F	FCC0FF	usr chksm	00..FF user
0080	FCC101	usr chksm	00..FF user
01D0	FCC3A1	bad sec total	00..max sector error only
01D1	FCC3A3	error trk #	00..max track error only
01D2	FCC3A5	error side #	00, 01 error only
01D3	FCC3A7	bad sector map	00..max sector error only
01F4	FCC3E9	buf begin	R/W data
03FF	FCC7FF	buf end	
COPS PB#4	FCDD81	FDIR	digital bit high = interrupt pending
Parallel PB#6	FCE901	DSKDIAG	digital bit low = busy

FCD

Pre issue requirements: DSKDIAG high  
gobyte zeroed

Post issue memory access: no special requirements  
gobyte zeroed after command validation

* tracks/sectors	tracks/sectors
0-3 / 0-15	17-1C / 0-11
4-A / 0-14	1D-22 / 0-10
B-10 / 0-13	23-29 / 0-F
11-16 / 0-12	2A-2D / 0-E

\*\*ALL VALUES ON THIS PAGE ARE HEX

Figure 6-61. FD Controller Seek Command



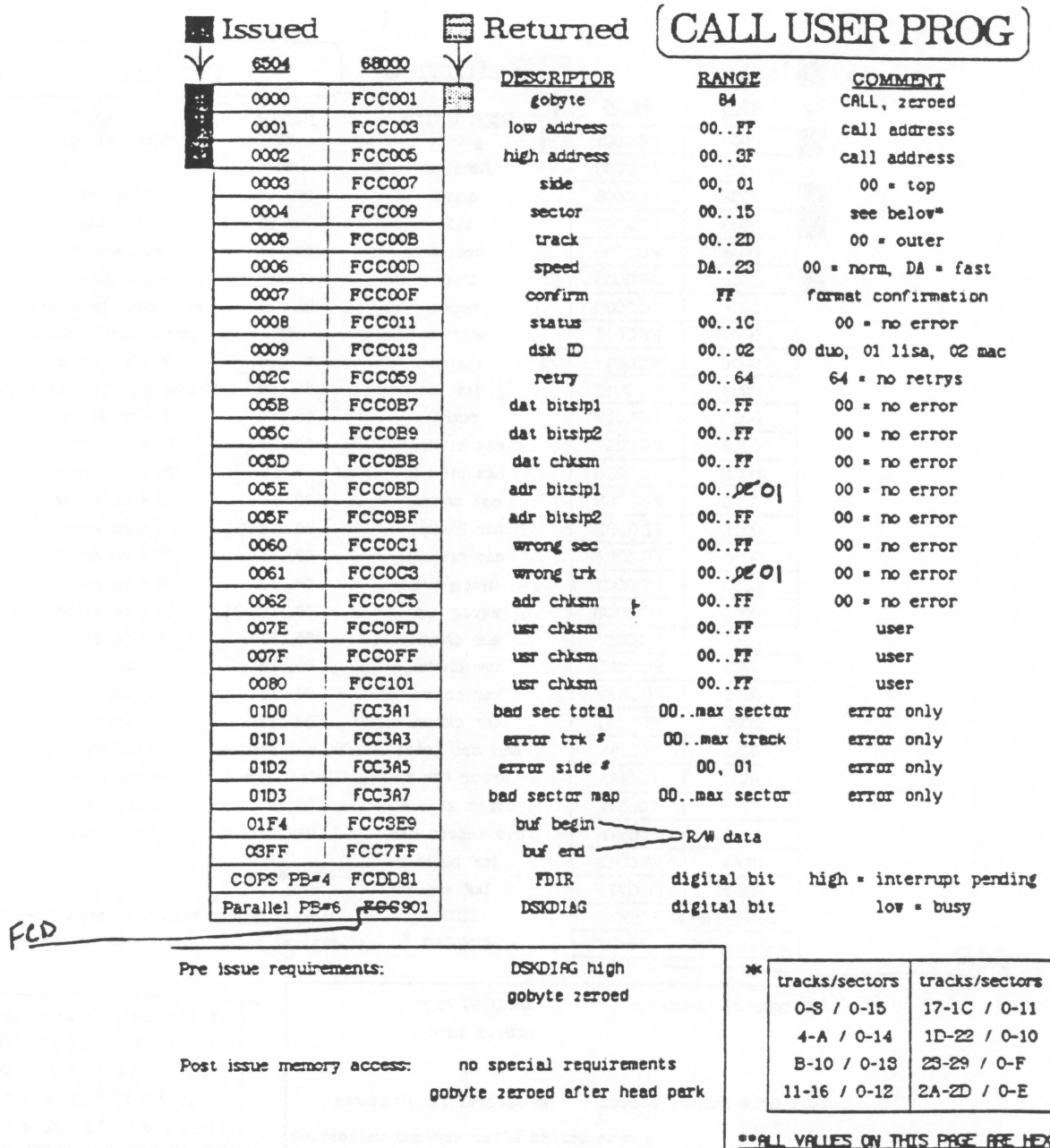


Figure 6-6m. FD Controller Call User Program Command

Issued		Returned		[CLEAR INT STAT]	
6504	68000	DESCRIPTOR	RANGE	COMMENT	
0000	FCC001	gobyte	85	CLEAR INTERRUPTS, zeroed	
0001	FCC003	mask	00..XAAAXBBB	X = dont care, B = top dr	
0002	FCC005	drive	00, 80	00 = top	
0003	FCC007	side	00, 01	00 = top	
0004	FCC009	sector	00..15	see below*	
0005	FCC00B	track	00..2D	00 = outer	
0006	FCC00D	speed	DA..23	00 = norm, DA = fast	
0007	FCC00F	confirm	FF	format confirmation	
0008	FCC011	status	00..1C	00 = no error	
0009	FCC013	dsk ID	00..02	00 duo, 01 lisa, 02 mac	
002C	FCC059	retry	00..64	64 = no retries	
005B	FCC0B7	dat bitslp1	00..FF	00 = no error	
005C	FCC0B9	dat bitslp2	00..FF	00 = no error	
005D	FCC0BB	dat chksm	00..FF	00 = no error	
005E	FCC0BD	adr bitslp1	00..001	00 = no error	
005F	FCC0BF	adr bitslp2	00..FF	00 = no error	
0060	FCC0C1	wrong sec	00..FF	00 = no error	
0061	FCC0C3	wrong trk	00..001	00 = no error	
0062	FCC0C5	adr chksm	00..FF	00 = no error	
007E	FCC0FD	usr chksm	00..FF	user	
007F	FCC0FF	usr chksm	00..FF	user	
0080	FCC101	usr chksm	00..FF	user	
01D0	FCC3A1	bad sec total	00..max sector	error only	
01D1	FCC3A3	error trk #	00..max track	error only	
01D2	FCC3A5	error side #	00, 01	error only	
01D3	FCC3A7	bad sector map	00..max sector	error only	
01F4	FCC3E9	buf begin	R/W data		
03FF	FCC7FF	buf end			
COPS PB#4	FCDD81	FDIR	digital bit	high = interrupt pending	
Parallel PB#6	FE901	DSKDIAG	digital bit	low = busy	

**\*\*ALL VALUES ON THIS PAGE ARE HEX**

Pre issue requirements:

DSKDIAG high

gobyte zeroed

Post issue memory access: no special requirements

gobyte zeroed after completion and error status update

bottom drv	MASK	top drv
0-3 / 0-15		17-1C / 0-11
4-A / 0-14		1D-22 / 0-10
B-10 / 0-13		23-29 / 0-F
11-16 / 0-12		2A-2D / 0-E

Figure 6-6n FD Controller Clear Interrupt Status Command



Issued		Returned		DRIVE ENABLE	
6504	68000	DESCRIPTOR	RANGE	COMMENT	
0000	FCC001	gobyte	86	DRIVE INT ENABLE, zeroed	
0001	FCC003	mask	00..XXXXXX	X = dont care, B = top	
0002	FCC005	drive	00, 80	00 = top	
0003	FCC007	side	00, 01	00 = top	
0004	FCC009	sector	00..15	see below*	
0005	FCC00B	track	00..2D	00 = outer	
0006	FCC00D	speed	DA..23	00 = norm, DA = fast	
0007	FCC00F	confirm	FF		
0008	FCC011	status	00..1C	00 = no error	
0009	FCC013	dsk ID	00..02	00 duo, 01 lisa, 02 mac	
002C	FCC059	retry	00..64	64 = no retrys	
005B	FCC0B7	dat bitsip1	00..FF	00 = no error	
005C	FCC0B9	dat bitsip2	00..FF	00 = no error	
005D	FCC0BB	dat chksm	00..FF	00 = no error	
005E	FCC0BD	adr bitsip1	00..0E01	00 = no error	
005F	FCC0BF	adr bitsip2	00..FF	00 = no error	
0060	FCC0C1	wrong sec	00..FF	00 = no error	
0061	FCC0C3	wrong trk	00..0E01	00 = no error	
0062	FCC0C5	adr chksm	00..FF	00 = no error	
007E	FCC0FD	usr chksm	00..FF	user	
007F	FCC0FF	usr chksm	00..FF	user	
0080	FCC101	usr chksm	00..FF	user	
01D0	FCC3A1	bad sec total	00..max sector	error only	
01D1	FCC3A3	error trk #	00..max track	error only	
01D2	FCC3A5	error side #	00, 01	error only	
01D3	FCC3A7	bad sector map	00..max sector	error only	
01F4	FCC3E9	buf begin	R/W data		
03FF	FCC7FF	buf end			
COPS PB#4	PCDD81	FDIR	digital bit	high = interrupt pending	
Parallel PB#6	PCCE901	DSKDIAG	digital bit	low = busy	

FCD

Pre issue requirements:

DSKDIAG high  
gobyte zeroedPost issue memory access: no special requirements  
gobyte zeroed after completion and error status update

* tracks/sectors	tracks/sectors
0-3 / 0-15	17-1C / 0-11
4-A / 0-14	1D-22 / 0-10
B-10 / 0-13	23-29 / 0-F
11-16 / 0-12	2A-2D / 0-E

\*\*ALL VALUES ON THIS PAGE ARE HEX

Figure 6-60. FD Controller Drive Enable Command

Issued		Returned		DRIVE DISABLE	
6504	68000	DESCRIPTOR	RANGE	COMMENT	
0000	FCC001	gobyte	87	DRIVE INT DISABLE, zeroed	
0001	FCC003	mask	00..XXXXXX	X = dont care, B = top	
0002	FCC005	drive	00, 80	00 = top	
0003	FCC007	side	00, 01	00 = top	
0004	FCC009	sector	00..15	see below*	
0005	FCC00B	track	00..2D	00 = outer	
0006	FCC00D	speed	DA..23	00 = norm, DA = fast	
0007	FCC00F	confirm	FF	format confirmation	
0008	FCC011	status	00..1C	00 = no error	
0009	FCC013	dsk ID	00..02	00 duo, 01 lisa, 02 mac	
002C	FCC059	retry	00..64	64 = no retrys	
005B	FCC0B7	dat bitslp1	00..FF	00 = no error	
005C	FCC0B9	dat bitslp2	00..FF	00 = no error	
005D	FCC0BB	dat chksm	00..FF	00 = no error	
005E	FCC0BD	adr bitslp1	00..0201	00 = no error	
005F	FCC0BF	adr bitslp2	00..FF	00 = no error	
0060	FCC0C1	wrong sec	00..FF	00 = no error	
0061	FCC0C3	wrong trk	00..0201	00 = no error	
0062	FCC0C5	adr chksm	00..FF	00 = no error	
007E	FCC0FD	usr chksm	00..FF	user	
007F	FCC0FF	usr chksm	00..FF	user	
0080	FCC101	usr chksm	00..FF	user	
01D0	FCC3A1	bad sec total	00..max sector	error only	
01D1	FCC3A3	error trk #	00..max track	error only	
01D2	FCC3A5	error side #	00, 01	error only	
01D3	FCC3A7	bad sector map	00..max sector	error only	
01F4	FCC3E9	buf begin	R/W data		
03FF	FCC7FF	buf end			
COPE PB#4	PCDD81	FDIR	digital bit	high = interrupt pending	
Parallel PB#6	PCD901	DSKDIAG	digital bit	low = busy	

FCD

Pre issue requirements:

DSKDIAG high  
gobyte zeroedPost issue memory access: no special requirements  
gobyte zeroed after completion and error status update

* tracks/sectors	tracks/sectors
0-3 / 0-15	17-1C / 0-11
4-A / 0-14	1D-22 / 0-10
B-10 / 0-13	23-29 / 0-F
11-16 / 0-12	2A-2D / 0-E

\*\*ALL VALUES ON THIS PAGE ARE HEX

Figure 6-6p. FD Controller Drive Disable Command

Issued		Returned		ROM WAIT	
6304	68000	DESCRIPTOR	RANGE	COMMENT	
0000	FCC001	gobyte	88	WAIT IN ROM UNTIL FLAGGED	
0001	FCC003	function	00		
0002	FCC005	drive	00, 80	00 = top	
0003	FCC007	side	00, 01	00 = top	
0004	FCC009	sector	00..15	see below*	
0005	FCC00B	track	00..2D	00 = outer	
0006	FCC00D	speed	DA..23	00 = norm, DA = fast	
0007	FCC00F	confirm	FF	format confirmation byte	
0008	FCC011	status	00..1C	00 = no error	
0009	FCC013	dsk ID	00..02	00 duo, 01 lisa, 02 mac	
002C	FCC059	retry	00..64	64 = no retrys	
005B	FCC0B7	dat bitsip1	00..FF	00 = no error	
005C	FCC0B9	dat bitsip2	00..FF	00 = no error	
005D	FCC0BB	dat chksm	00..FF	00 = no error	
005E	FCC0BD	adr bitsip1	00..0E01	00 = no error	
005F	FCC0BF	adr bitsip2	00..FF	00 = no error	
0060	FCC0C1	wrong sec	00..FF	00 = no error	
0061	FCC0C3	wrong trk	00..0E01	00 = no error	
0062	FCC0C5	adr chksm	00..FF	00 = no error	
007E	FCC0FD	usr chksm	00..FF	user	
007F	FCC0FF	usr chksm	00..FF	user	
0080	FCC101	usr chksm	00..FF	user	
01D0	FCC3A1	bad sec total	00..max sector	error only	
01D1	FCC3A3	error trk #	00..max track	error only	
01D2	FCC3A5	error side #	00, 01	error only	
01D3	FCC3A7	bad sector map	00..max sector	error only	
01F4	FCC3E9	buf begin	R/W data		
03FF	FCC7FF	buf end			
COPS PB#4	FCDD81	FDIR	digital bit	high = interrupt pending	
Parallel PB#6	FCDD91	DSKDIAG	digital bit	low = busy	

FCD

Pre issue requirements:

DSKDIAG high  
go byte zeroed

Post issue memory access:

no special requirements  
gobyte zeroed after head park

6304 waits until 69, 96 sequence in gobyte

* tracks/sectors	tracks/sectors
0-S / 0-15	17-1C / 0-11
4-A / 0-14	1D-22 / 0-10
B-10 / 0-13	23-29 / 0-F
11-16 / 0-12	2A-2D / 0-E

\*\*ALL VALUES ON THIS PAGE ARE HEX

Figure 6-6q. FD Controller ROM Wait Command

Issued		Returned		GO AWAY	
6504	68000	DESCRIPTOR	RANGE	COMMENT	
0000	FCC001	gobyte	89	LOOP IN ROM PERMANENTLY	
0001	FCC003	function	00		
0002	FCC005	drive	00, 80	00 = top	
0003	FCC007	side	00, 01	00 = top	
0004	FCC009	sector	00..15	see below*	
0005	FCC00B	track	00..2D	00 = outer	
0006	FCC00D	speed	DA..23	00 = norm, DA = fast	
0007	FCC00F	confirm	FF	format confirmation	
0008	FCC011	status	00..1C	00 = no error	
0009	FCC013	dsk ID	00..02	00 duo, 01 lisa, 02 mac	
002C	FCC059	retry	00..64	64 = no retries	
005B	FCC0B7	dat bitslp1	00..FF	00 = no error	
005C	FCC0B9	dat bitslp2	00..FF	00 = no error	
005D	FCC0BB	dat chksm	00..FF	00 = no error	
005E	FCC0BD	adr bitslp1	00..0201	00 = no error	
005F	FCC0BF	adr bitslp2	00..FF	00 = no error	
0060	FCC0C1	wrong sec	00..FF	00 = no error	
0061	FCC0C3	wrong trk	00..0201	00 = no error	
0062	FCC0C5	adr chksm	00..FF	00 = no error	
007E	FCC0FD	usr chksm	00..FF	user	
007F	FCC0FF	usr chksm	00..FF	user	
0080	FCC101	usr chksm	00..FF	user	
01D0	FCC3A1	bad sec total	00..max sector	error only	
01D1	FCC3A3	error trk #	00..max track	error only	
01D2	FCC3A5	error side #	00, 01	error only	
01D3	FCC3A7	bad sector map	00..max sector	error only	
01F4	FCC3E9	buf begin	R/W data		
03FF	FCC7FF	buf end			
COPS PB=4	FCDD81	FDIR	digital bit	high = interrupt pending	
Parallel PB=6	FCDD901	DSKDIAG	digital bit	low = busy	

*FCD*

Pre issue requirements:	DSKDIAG high gobyte zeroed
Post issue memory access:	no special requirements

* tracks/sectors	tracks/sectors
0-8 / 0-15	17-1C / 0-11
4-A / 0-14	1D-22 / 0-10
B-10 / 0-18	23-29 / 0-F
11-16 / 0-12	2A-2D / 0-E

**\*\*ALL VALUES ON THIS PAGE ARE HEX**

Figure 6-6r. FD Controller Go Away Command

Execution of the macro commands received from the CPU are under the control of the 6504. Upon completion of the operation, the status of the controller is available in the I/O control block for interrogation by the CPU. Figure 6-7 shows the general flow of macro command transfer and execution.

Figure 6-7. Floppy-Disk Macro Command Flowchart

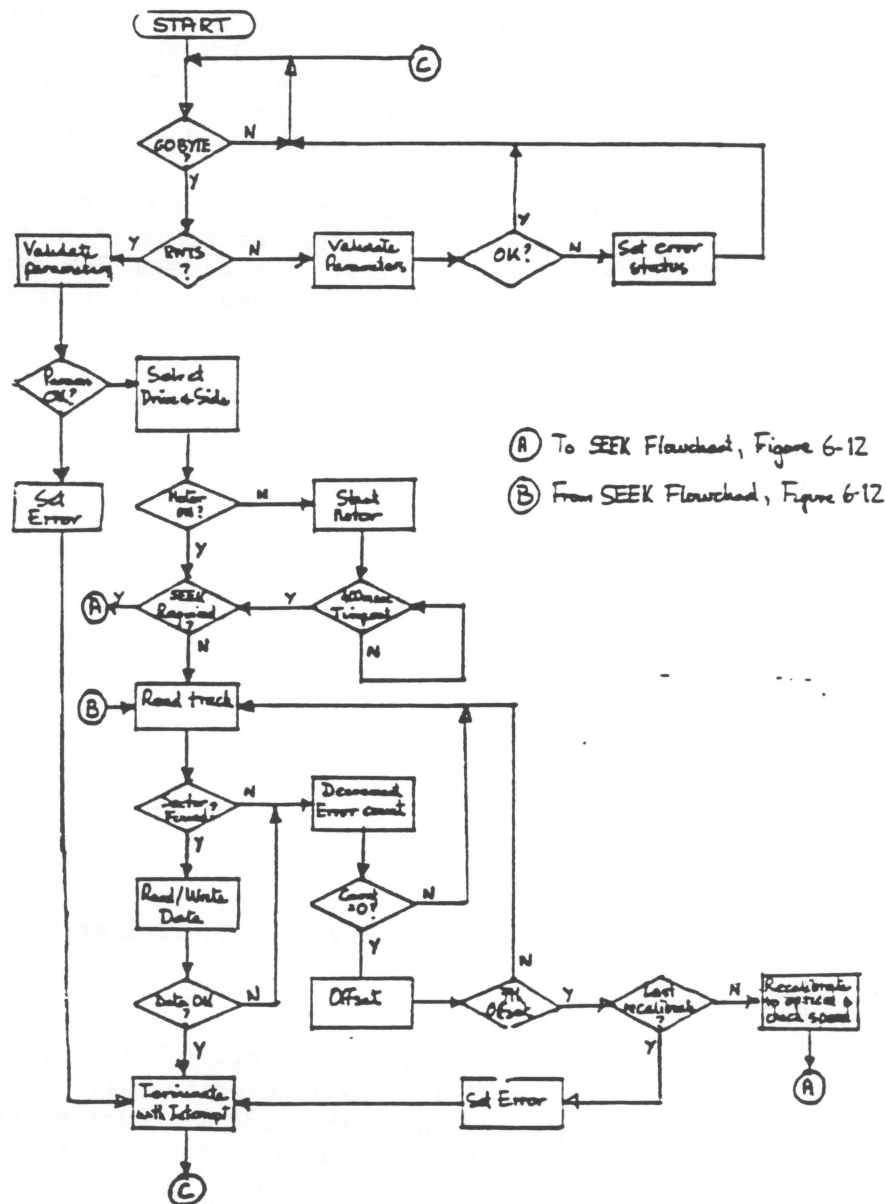


Figure 6-7. FD Macro Command Flowchart

### 6.3.1 Disk Processor Operation

The 6504 processor executes code from the local ROM storage within the floppy-disk controller. It is routed to the routines contained therein on the basis of the macro command placed in the I/O control block of the local RAM by the 68000.

The 6504 is also responsible for the encoding and decoding of the data for transfer to and from the floppy-disk drive. The command RAM available to the 6504 is logically divided as shown in Figure 6-8.

Figure 6-8. Floppy-Disk Controller I/O Block

The floppy-disk controller RAM is shown in detail in Figure 6-3. The read/write shared RAM is initialized by the 6504 at power-on time. The status block is used by the 6504 to indicate the result of execution of a macro command. The internal I/O block is used by the 6504 for internal flags indicating progress in execution of an operation. Variables used by the 6504 include both local and variable intermixed. The parameter storage is a space available to the 68000 and is not accessed by the 6504. The 6504 stack fills the space assigned to it from the high-order position downwards. The I/O Buffer is a space of 524 bytes used to transfer data between the floppy disk and the 68000.

In order to proceed with disk operations, the 6504 inhibits interruption by the 68000. This enables the timing routines that are performed by the 6504

0000	GO BYTE		
0001	COMMAND	ADR(L)	MASK
0002	DRIVE	ADR(H)	
0003	SIDE		
0004	SECTOR		
0005	TRACK		
0006	SPEED		
0007	FORMAT CONFIRM		
0008	ERRSTAT		
0009	DISK ID (0, 1 or 2)		
000A : : : 000F	Reserved for Future Use.		

Figure 6-8. FD Controller I/O Block



to be performed in real time until the operation is complete and termination status is available to be interrogated by the 68000.

The 6504 firmware always executes in one of three possible states:

- \* Waiting for a macro command from the 68000
- \* Checking the status of the drives
- \* Executing a macro command.

When the controller is waiting for a command, the 6504 uses a three-byte counter to control how long the motors are left on after the last macro command executed.

When the two low-order bytes decrement to zero, the status of the disk is checked. This occurs about once per second. The high byte of the counter is directly modifiable by the 68000.

After disk status has been checked, the GOBYTE is examined to see if a macro command has been loaded and is awaiting execution. If the high-order bit is set, for example, if the GOBYTE contains a negative number, this indicates a command waiting. The counter is decremented at approximately 10 microsecond intervals. At a count of zero, the disk drive motors are stopped.

The disk status is checked approximately once each 2/3 second. It is not checked more frequently because the routine takes time and blocks any response to the 68000 while it is executing. It also involves turning on the phi0 line to the disk motor, which causes noise.

Both drives are checked for disk-in-place. Should interrupts be enabled on a drive that changes this status at this time, the 68000 is interrupted to inform it of the fact. The 68000 must clear the interrupt to issue the command that clamps the disk. On a drive for which interrupts are not enabled, the disk-in-place condition causes the disk to be clamped automatically.

If the status shows the eject button was pushed, nothing happens if no disk-in-place condition exists. Otherwise, should interrupts be enabled, an interrupt is sent to the 68000. If interrupts are disabled, the disk is unclamped automatically.

### 6.3.2 Disk Macro Commands

The 6504 detects that the CPU has transmitted a macro command to the floppy-disk controller by monitoring the high-order bit of the GOBYTE in the I/O block, 10B, for a 1 (one).

Upon detecting this, the first eight bytes of the IOB are copied to the internal I/O block (IIOB). Before any execution, interrupts are checked to see if any are pending. If there are, the Clear Interrupt Status command is allowed to execute.

The macro commands are divided into two classes, depending on whether they do or do not interrupt the 68000 upon completion. Those that do interrupt generally require extensive processing by the 6504 and the 68000 is locked out of shared memory for the duration of instruction execution. Once the class of the command has been determined, the GOBYTE is cleared and the command parameters checked for errors. If an error is detected the cause is placed in the ERRSTAT byte and the command is aborted.

The macro commands that can be received follow. The details of the commands and their associated parameters are shown in Figure 6-6 above.

#### READ/WRITE DATA

Operations that involve an implied seek operation. The I/O command block therefore contains information on the desired drive, side, track, and sector. The information returned to the 68000 includes the status upon command completion, a pointer to the buffer containing data read, and a counter for the number of retries that were required.

#### UNCLAMP

The operation that releases the disk currently inserted in the drive so that it can be removed. The input data states which drive, 00 or 80, and the 68000 is interrupted by means of the FDIR signal.

#### CLAMP

The operation that clamps a disk that has been inserted in a drive for which interrupts are

enabled.

#### FORMAT and FORMAT TRACK

Operations that overwrite all data and markers on the given track(s). They provide status information, which includes the location of bad sectors that could not be formatted correctly.

#### VERIFY and VERIFY TRACK

Operations that attempt to read track(s) and inform the 68000 where errors are present in a manner similar to the FORMAT. However, no write operation is performed.

#### READ BRUTE FORCE (Ignore Checksum)

Identical to a READ operation, except that agreement of the data with the three checksum bytes written at the end of each sector is ignored. It enables partly erroneous data to be read.

#### WRITE BRUTE FORCE (User Checksum)

Identical to the WRITE operation, except that the command supplies the three checksum bytes to be written at the end of the sector. The controller does not attempt to write the bytes that it would normally generate. Ordinarily, data thus written can only be read without error by a READ BRUTE FORCE (Ignore Checksum) operation.

#### SEEK

The operation that moves the selected head to the track required, but does not transfer any data.

#### CALL USER PROGRAM

The operation that initiates a routine that has been down-loaded to the 6504 RAM area.

#### CLEAR INTERRUPT STATUS

The operation that clears the interrupt status to

the data output.

#### DRIVE ENABLE/DISABLE

The commands used to control the availability of either of the two disk drives to the 68000.

#### ROM WAIT

The command that continually reads RAM, looking for a two-byte sequence of 69 and 96. Once these are found, the 6504 performs a cold start of the 6504. The purpose of this operation is primarily to remove the 6504 from the RAM area while diagnostics are being performed.

#### GO AWAY

The command that causes the 6504 to loop in ROM, never accessing RAM or I/O. It is used to prevent an erroneous WRITE operation to parameter memory when the Lisa is turned off. Note that once issued, this command cannot be stopped without resetting the Lisa.

These operations result in a number of possible status codes being returned to the 68000 via the status byte, which indicates an error when it is non-zero. In addition, error counters give the number of occurrences of specific errors during the performance of a given operation.

The floppy-disk controller attempts to retry an operation on the disk, which results in an error condition. This is attempted repeatedly until the error count is exceeded. Retries involve moving the head off track in increments of 1/8th of a track and recalibrating the head using the optical sensor on the drive.

The meaning of the possible error codes and the error counters is shown in Figure 6-9.

ERROR CODES	SIGNIFICANCE
01 (hex)	Invalid Command
02	Invalid Drive
03	Invalid Sector
04	Invalid Side
05	Invalid Track
06	Invalid Clear Mask
07	No Disk
08	Drive Not Enabled
09	Interrupts Pending
0A	Invalid Format Confirmation
0B	ROM Selftest Failure
0C	Unexpected IRQ or NMI
14	Write Protect Error
15	Unable to Verify
16	Unable to Clamp
17	Unable to Read
18	Unable to Write
19	Unable to Unclamp
1A	Unable to Find Calibration
1B	Unable to Adjust Speed
1C	Unable to Write Calibration

ERROR COUNTER ADDRESS	SIGNIFICANCE
005B (hex)	Bitslip Error Count (data field start header)
005C	Bitslip Error Count (data field trailer)
005D	Checksum Error Count
005E	Bitslip Error Count (ID field start header)
005F	Bitslip Error Count (ID field trailer)
0060	Wrong Sector
0061	Wrong Track
0062	Header Checksum Error

Figure 6-9. Floppy-Disk Controller Error Codes

### 6.3.3 Data Encoding/Decoding

Data transferred to the floppy-disk controller by the 68000 for storage on the drives is not written directly onto the disk. The data are encoded in the

following manner before it is stored:

- \* Three 8-bit bytes become four encoded bytes,
- \* The MSB of each encoded byte on the disk begins with a one,
- \* No more than two zero bits occur consecutively.

The data are encoded under control of the 6504 after it has been placed in the buffer by the 68000, as shown in Figure 6-10.

—→ *ll* #'s

GCR Codeword Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	96	97	9A	9B	9D	9E	9F	A6	A7	AB	AC	AD	AE	AF	B2	B3
10	B4	B5	B6	B7	B9	BA	BB	BC	BD	BE	BF	CB	CD	CE	CF	D3
20	D6	D7	D9	DA	DB	DC	DD	DE	DF	E5	E6	E7	E9	EA	EB	EC
30	ED	EE	EF	F2	F3	F4	F5	F6	F7	F9	FA	FB	FC	FD	FE	FF

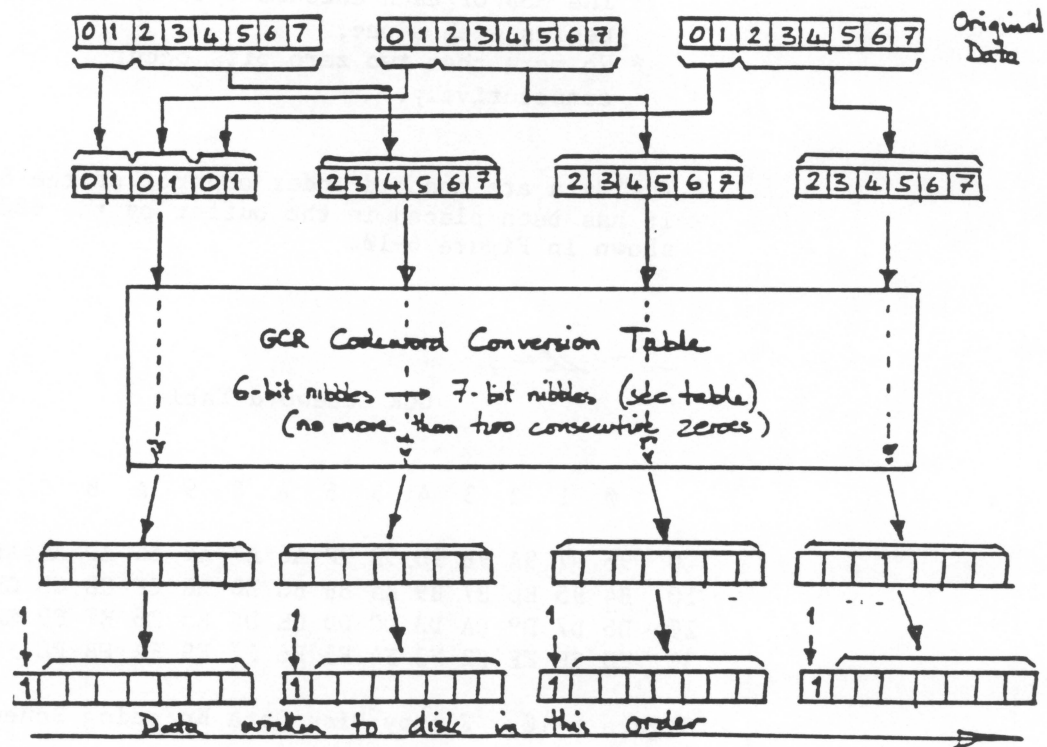
Figure 6-10. Floppy-Disk Data Encoding Scheme

The MSB of each encoded byte must be a one so the drive state machine can identify byte boundaries when the data are being read from disk. The requirement that no more than two zeroes occur consecutively permits data to be written without intervening clock bits. Transitions from the data bits themselves occur frequently enough to permit the state machine to remain synchronized with the data being read.

The data are encoded from groups of three 8-bit bytes into four encoded bytes by a table lookup operation. It so happens that within the 128 possible codings of the other seven bits written to the disk within one byte, there are more than 64 codings that satisfy the requirement of no more than two consecutive zeroes.

#### 6.3.4 Disk Formatting

A number of codings that are not used in data translation are used to indicate a number of special



GCR Codeword Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	96	97	9A	9B	9D	9E	9F	A6	A7	AB	AC	AD	AE	AF	B2	B3
10	B4	B5	B6	B7	B9	BA	BB	BC	BD	BE	BF	CB	CD	CE	CF	D3
20	D6	D7	D9	DA	DB	DC	DD	DE	DF	E5	E6	E7	E9	EA	EB	EC
30	ED	EE	EF	F2	F3	F4	F5	F6	F7	F9	FA	FB	FC	FD	FE	FF

Figure 6-10. FD Data Encoding Scheme

Floppy-Disk



codes, such as headers within the format and speed synchronization bytes.

The number of sectors and tracks are shown in Figure 6-11. Note that the number of sectors is dependent on the track. This is due to the technique of varying the rotational speed of the drive to allow for the increased density possible on the outer tracks of a disk.

→ ll #5

Tracks	Sectors	r.p.m.	Speed Code
0-3	22	218.3	DC (hex)
4-10	21	228.7	CC
11-16	20	240.1	B8
17-22	19	252.7	A4
23-28	18	266.8	8E
29-34	17	282.5	75
35-41	16	300.1	59
42-45	15	320.1	38

Figure 6-11. Lisa Disk Format

Tracks 0 through 45 are normal data tracks. The number of sectors contained on each track depends on the track position as shown. This is due to differing lengths of each track. The speed of disk rotation is dynamically changed by the floppy-disk controller to take advantage of this.

Track -1 is used as a speed synchronization track. It is never written except during disk formatting. The 6504 makes use of this track to adjust the speed of rotation to within 0.4% of the desired speed.

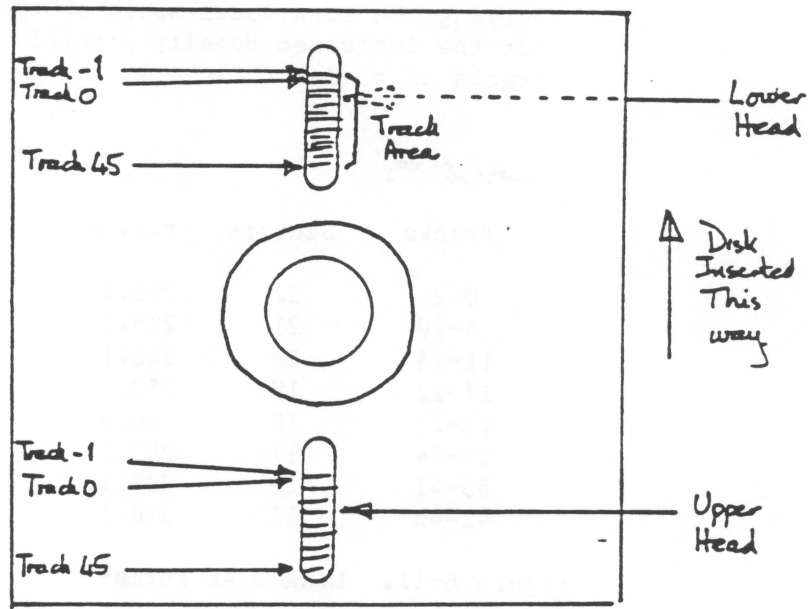
Each sector is divided into four distinct fields, as described below:

- \* Header sync field
- \* Header field
- \* Data sync field
- \* Data field.

#### HEADER SYNC FIELD

Contains a pattern of ones and zeroes that permits the disk state machine to synchronize with the data coming from disk. It consists of the





Tracks	Sectors	r.p.m.	Speed Code
0-3	22	218.3	DC (hex)
4-10	21	228.7	CC
11-16	20	240.1	B8
17-22	19	252.7	A4
23-28	18	266.8	8E
29-34	17	282.5	75
35-41	16	300.1	59
42-45	15	320.1	38

Figure 6-11. LISA Disk Format

following pattern:

32 "Self Synch" FFs  
 10 bytes A9 Speed Synchronization  
 (only before Sector #0)

#### HEADER FIELD

Uniquely identifies the sector on the disk, using the following pattern:

D5	)	
AA	)	Header field identification
96	)	
1 byte		TRACK identification number
1 byte		SECTOR identification number
1 byte		SIDE identification. 00 or 01
1 byte		VOLUME identification. May be one of 00=AppleII or ///, 01=Lisa, 02=Mac
1 byte		CHECKSUM coding. This is formed by XOR'ing the previous four bytes
DE	)	
AA	)	"Bit slip" marks
1 byte		Pad byte where head is turned off

#### DATA SYNC FIELD

Similar in purpose to the header sync field. It has the pattern:

5 "Self Synch" FFs

#### DATA FIELD

Contains the data written to the drive. Normally this is the only field written to except during a format operation. It has the pattern:

D5	)	
AA	)	Data marks that identify a data field
AD	)	
1 byte		Sector number
524 bytes		User data
3 bytes		Checksum formed by adding data
DE	)	
AA	)	"Bit slip" marks
1 byte		Pad byte where head is turned off

Note that the format is given in terms of the bytes handled by the 68000. The actual number of

bytes written to the disk is greater, since three normal bytes map into four bytes on the disk, as given by Figure 6-10.

Self synch refers to a technique whereby two zeroes are inserted between each synch byte written to the disk. The state machine requires no more than four self-synch fields, 8 ones and 2 zeroes, to synchronize its read operation with disk rotation.

### 6.3.5 Disk State Machine Operation

The state machine that controls the flow of data across the drive interface is shown in Figure 6-5 above.

There are five external inputs to the state machine. These are the RDA, WRD, and SNS lines from the disk, and the A2 and A3 control lines from the control outputs of the LS259 shown at C-2 on sheet 4. The latter two are used to control the command that the state machine is currently executing. The four possible commands are:

- \* Sense
- \* Read
- \* Write
- \* Write Load.

In addition, there are two signals that are input to the state machine ROM. These are the QA output of the shift register, which provides the data stream to be written to the disk, and the output of the edge detector at D-1, which presents the data transitions to the ROM as ones.

### SPINDLE MOTOR SPEED CONTROL

During normal operation, the disk drive controls the rotation by means of an integral speed control circuit. When the drive performs a seek operation outside of the current range of tracks, the speed is adjusted according to the values given in Figure 6-11. Disk speed is maintained to within 3% of that desired.

DRIVE REZERO and SEEK

Performed in a similar manner to ??? in that they both involve carriage motion. The four lines  $\phi_0$ - $\phi_3$  are manipulated to cause the carriage motor to step the required amount of tracks. In the case of a rezero, a sense operation determines whether the optical recalibration line is asserted to indicate that the carriage is at the known position of track -1.

Carriage movement is defined by the order in which the  $\phi$  lines are manipulated. Thus a step through  $\phi_3, \phi_2, \phi_1$ , and  $\phi_0$  moves the heads toward the calibration point, towards the front of the Lisa, by one half track. Stepping through  $\phi_0, \phi_1, \phi_2$ , and  $\phi_3$  moves the heads away from the calibration point by one half track.

In order to minimize the positioning error from mechanical tolerances seeks are always completed by moving the carriage in the same direction, towards the calibration point, for the last half track, which is four increments. This means that seek direction is reversed during a seek away from the calibration point.

Carriage movement is performed in increments of 1/8th of a track. Allowance is made for head inertia by having three distinct time durations of each  $\phi$  state before proceeding to the next. An example of a seek is shown in Figure 6-12.

Figure 6-12. Seek Flowchart

WRITE and WRITE LOAD

These commands operate together to provide the write function to the drive. Each command consists of 16 states in the state machine. Each state is passed through in one clock cycle of 250 ns.

Since the bit time on the disk is 2 microseconds, the 16 states correspond to two bit times. The

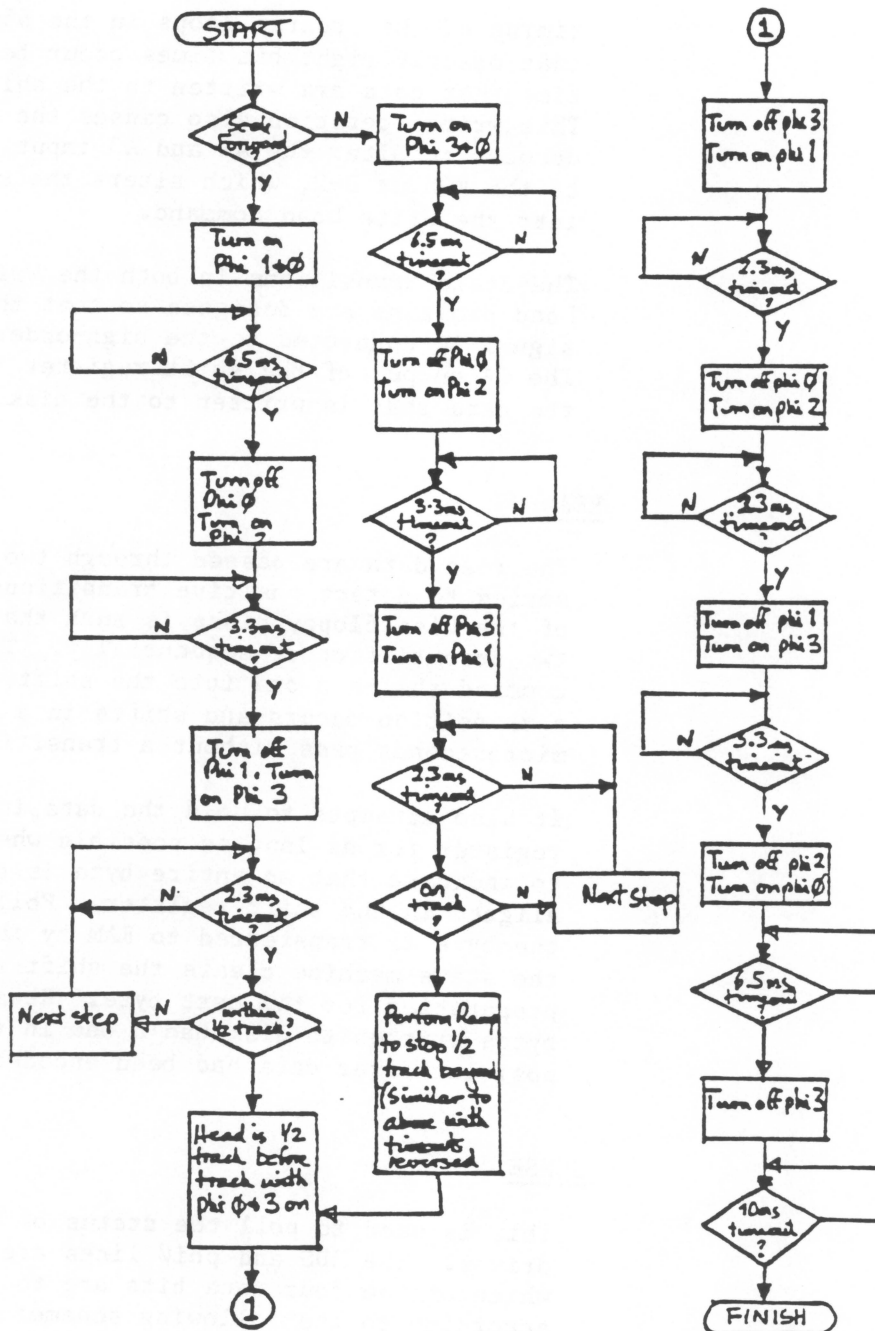


Figure 6-12. Seek Flowchart

timing of the program loops in the 6504A is such that exactly eight bit times occur between each time that data are written to the shift register. This write operation also causes the upper control decoder to alter the A2 and A3 input configuration to the ROM at D-2, which alters the state machine into the Write Load command.

The state transitions in both the Write and Write Load programs are designed so that the WRD signal is connected to the high-order state bit. The QA output of the shift register thus controls the data that is written to the disk.

### READ

The read data are passed through two flops in series to detect positive transitions. The format of the Lisa floppy disks is such that no more than two zero bits occur sequentially. The read command shifts a one into the shift register when a transition occurs and shifts in a zero when 2 microseconds pass without a transition.

It also attempts to hold the data in the shift register for as long as possible when QA goes high to indicate that an entire byte is correctly aligned in the shift register. Following this, the byte is transferred to RAM by the 6504, and the state machine clears the shift register in preparation for the next byte. Recall that all bytes written to disk had a one in the MSB position after data had been encoded.

### SENSE

This is used to poll the status of the disk drives. The HDS and phi0 lines are used to select which of the four data bits are to be read according to the following scheme:

HDS	phi0	Status Data
0	0	Write Protect
0	1	Optical Recalibration
1	0	Eject Button
1	1	Disk in Place

After the control lines have been configured to sense the required data, the command is initiated

by setting the A2 and A3 inputs to 1@2. The MSB of the shift register contains the sense bit. All 16 possible states in the sense command are identical and cause a shift right operation.

#### 6.4 The Serial I/O Controller

The I/O Board contains two serial interface ports, A and B, that are controlled by a single serial I/O controller. The software interface to this controller is discussed in Chapter 2. The A-port conforms to the RS232A specification; the B-port conforms to RS422.

The control logic is shown on the right side of sheet 3 of schematic 050-4008. Refer to the block diagram of the board in Figure 6-1 for a view of the hardware context in which the serial controller operates. Programming of the ports is discussed in subsection 3.5.1.

##### 6.4.1 The Serial Port Controller

The serial ports are controlled by an 8530 dual serial-port controller. A discussion of the controller can be found in the 8530 user manual.

The controller connects to the internal I/O board D bus. It is selected when the VMA signal coincides with AS. The additional selection is performed on the read and write inputs, which are derived from the port decoding logic at B-3 on sheet 2.

The WSIO/ signal enables a write to the controller, while an RSIO/ enables a read from it. These signals originate in the controller decode logic at A-3 on sheet 2. A configuration of address lines A11-A9 of 001 selects the serial-port controller. The READ line distinguishes a read from a write.

The two low-order address bits of the system address bus A1 and A2 are used to select between the two ports and to define whether control information or data are being transferred.

The PCLK device clock is derived from the 4 M, 4 MHz, clock signal at A-3 on sheet 2. This is in turn derived from the 16 MHz clock from the floppy-disk controller.

The controller interrupts the processor through the RSIR/ signal on the system bus. Selection of registers and functions within the controller is



made by means of the D/C/ and A/B/ pins. The first indicates whether data or command information is present on the bus lines. The second selects between the A and B ports. Address lines A1 and A2 enable software to control this.

#### 6.4.2 The Serial Ports

The two serial ports are provided in the form of two 25-pin DB connectors, as described in subsection 3.5.1.

The logic interface is shown on sheet 3 of the schematics at A-1 through D-1. The signals that correspond to port A are in the upper section of the page and have signal names ending in "1." The signal names and their function correspond to standard RS232A nomenclature. The 1488 and 1489 devices convert the TTL signals from the controller into the + or -12 V levels of the interface and vice-versa.

Port B signals are in the lower part of the page. They provide an RS422-standard interface through use of the 26LS30 and 26LS32 devices.

#### 6.4.3 Baud Rate Generation

Selection of the baud rate used in either of the two ports is performed internally by the 8530 with the aid of the 3.68 MHz oscillator between the SYNB and TRXCB lines. Refer to the 8530 user manual for details on how baud rates are selected.

#### 6.4.4 Serial Port Operation

The two serial ports operate independently of one another. For details, refer to the serial-port controller user manual.

### 6.5 Parallel Port Controller

The parallel port is used to interface a parallel peripheral to the Lisa. A typical application is in the attachment of an Apple ProFile hard disk drive.

The interface is discussed operationally in Chapter 2 of this manual. The logic is shown on sheet 3 of schematic 050-4008.

The hardware consists of a 6522 I/O port device and some

associated logic. The 6522 is a 68000-type peripheral and is described in the 6522 data sheet.

#### 6.5.1 68000 Bus Interface for the Parallel-Port Controller

This is provided by the 6522. The I/O Board internal D-bus connects to the data lines of the 6522, while addressing within the 16 locations contained in the port controller is performed with the system bus address lines A3 through A6.

Device selection is performed by having both the DSKPT/ and VMA signals true. The former is provided by the I/O decode at B-3 on sheet 1, while the latter indicates that the processor is aware that it is communicating with a 68000-type device and has modified the bus signals accordingly. Refer to the 68000 user manual for a discussion of the VMA signal.

The resulting address map for the port is shown in Figures 2-11 and 2-12.

6522 reset is performed at the same time as Lisa reset by means of the RESET/ signal at D-4, which originates on the system bus.

The 6522 interrupts the 68000 by means of the IOIR/ line, which is wired-ORed with the floppy-disk controller interrupt (FDIR/ ) originating at B-1 on sheet 4.

6522 clocking is performed by the E signal from the system bus, which is connected to the phi2 input and provides a 68000-type compatible clock.

#### 6.5.2 Parallel Port Interface

The 6522 provides two 8-bit ports, the A port and the B port, for use in the interface itself. The A port is used as the 8-bit parallel data interface, while the B port is used for control lines.

The LS280 at D-3 is used to check parity as data are being read from the interface into the Lisa. The result of the check is latched in the LS109 JK flop at D-3. The Reset input is presented as PRES/ to the keyboard port at C-3 on sheet 2 to permit the flip-flop to be reset. The Q output is on the 6522. The second LS280 at D-3 is used to generate parity.

The lines on the B port are used as control and status lines on the interface as follows:

BSY connects to PB1 and indicates to the 6522 that the interface is busy. It also connects to CA1 to provide the data interrupt input.

Open Cable Detect (OCD) connects to the PB0 line of the 6522. This signal is normally held low at the peripheral end to signify that the cable is connected. The Lisa monitors this line and assumes that no transfer can be performed when it is high.

The signal that is driven by PB2 is used as an enable to the drivers' interface. When it is low, both the LS245 8-bit data transceiver and the LS244 control line drivers are enabled.

DR/W/ is output from the PB3 line to the peripheral as an indication of the direction of data transfer. It is also used to define the direction of the data transceiver's drivers.

CMD/ is output to the peripheral from the PB4 line.

DSKDIAG is not input from the peripheral but originates in the floppy-disk controller section at B-1 on sheet 4. It indicates on PB6 that the 68000 is operating the interface in diagnostic mode.

The port control lines of the 6522 are used as follows:

CA1 is used to monitor the state of the BSY line from the peripheral and to strobe the data on the A port accordingly.

CA2 is used to provide the Peripheral Strobe (PSTRB/ ) signal to the interface, which indicates that either data are available for a write or data has been received for a read operation.

CB2 is used to monitor the parity status of the interface.

### 6.5.3 Parallel Port Operation

Operation of the parallel port is under software control and lies outside the scope of this manual. Refer to software documentation and Chapter 2 of this manual for programming information on this port.

### 6.5.4 Parallel Port Timing

Timing on the parallel port is programmable and a function of software. It therefore lies outside the scope of this manual. Refer to the data sheet for a discussion of the timing limitations on the 6522.

## 6.6 The Keyboard/Mouse Controller

The interface to the keyboard and the mouse is implemented by means of a 6522 VIA peripheral port device and a COPS single-chip controller. It is also used to provide software control of the power on-off function and to provide a real-time-clock.

Details on the 6522 can be found in the data sheet. Refer to the COPS user manual for information on the COPS. The logic that comprises the interface controller is shown on sheet 2 of schematic 050-4008 in Appendix C. Programming of the controller is discussed in subsection 2.5.4.

### 6.6.1 68000 Bus Interface for the Keyboard/Mouse Controller

This interface is implemented with the 6522. The internal I/O board D-bus connects to the D0-D7 data lines. Selection among the 16 internal register destinations is performed by the A1 through A4 address lines from the system bus.

The device is selected when both the VMA signal on the system bus and the I/O decode output at pin 7 of device U4E are asserted. This indicates that the processor board is performing an access to the keyboard/mouse controller in a 6800 compatible cycle.

The address map of the controller as seen by the software is shown in Figure 2-15.

The device is clocked by the E signal and reset by the RESET/ line.

### 6.6.2 The COPS Processor

The COP421 shown at D-3 on sheet 2 is described in detail in the COPS user manual. As well as controlling data flow from the keyboard and mouse, the COP421 is responsible for maintaining the Time of Day clock. To do that it requires a power supply that is independent of the main supply; one that remains on even when the Lisa is turned off, as described in the next subsection.

The COPS connects to the A port of the 6522, with the CA1 and CA2 control signals of the 6522 being attached to the S0 and SI lines of the COPS. In addition, the port PB6 control output signal connects with D3.

The other lines from the COPS are used as follows:

SK is the output to the keyboard. In conjunction with the D2 line, it is used to send a synchronization pulse to the keyboard to initiate data transfer.

G0 and G1 are the multiplexed data inputs from the keyboard and the mouse.

D1 and D2 are the select signals that are used to control the data multiplexer which provides keyboard and mouse data on G0 and G1.

D0 is used to switch the Lisa on and off under control of firmware resident in the COPS.

G2 is used to interrupt the processor by means of the NMI/ , non-maskable interrupt, at B-2 on sheet 2, which is presented to the processor board via the system bus, and to detect a power failure via D5 at B-2 on sheet 2.

G3 is used to sense the state of the on-off switch on the lower-left of the Lisa cabinet.

CK1 and CK2 are inputs to the COPS oscillator from timing circuit at D-2. The clock is crystal-controlled to permit the time of day to be kept accurately.

RST is the COPS internal reset input which is used to perform a power-on reset in the COPS should its own power have failed for any reason.

### 6.6.3 Keyboard/Mouse Interface

Keyboard data are input on the KBD line. This line is pulled low by the COPS generating a SYNC pulse to signal the keyboard to send data if it has any. This is done via the COPS SK output with the D2 output being asserted simultaneously.

The keyboard responds with an ACK pulse on the KBD line, followed by a serial 8-bit byte which indicates the code for a key pressed or released. The keycodes are outlined in Figure 2-14. If no key has changed its state, no ACK pulse is sent.

The mouse interface consists of the LS153 dual 4-to-1 multiplexer at D-2. The selection of data to be input to the COPS is performed by configuring the D1 and D2 outputs from the COPS.

The COPS polls the signal states by reading the signals input to the multiplexer. The movement of the mouse is detected by pulse edges on the relevant direction lines. The three inputs SW0-SW2 reflect the state of up to three switches on the mouse. In the current Lisa only one mouse switch is present and is connected to the SW0 line.

The data presented to the COPS is selected as shown in Figure 6-13. Refer to Chapter 8 for a discussion of the keyboard side of this interface.

Figure 6-13. Keyboard Data Format and Timing

#### 6.6.4 Software On-Off/Reset Logic

The software on-off switch is sensed through the PWRSW/ signal. In addition, pulling the RESET/ signal high when the Lisa is turned off pulls the PWRSW/ signal low and turns the Lisa on. Note that this can be also accomplished by pulling RESET/ to +5STBY with a 100 ohm resistor.

The COPS can turn the Lisa on or off, via the ON line. The COPS may be programmed to turn the Lisa on under control of the real-time-clock (RTC). The RTC is not capable of turning the computer off.

#### 6.6.5 Other Control Lines

The keyboard and mouse controller makes use of the A port of the 6522. The B port is used to provide an interface to the Lisa for several control lines that would otherwise require additional hardware to implement.

The use of the B port lines is as follows:

PB0 is used to reset the keyboard under software control.







PB1-PB3 are used for output of a digital value to control the speaker volume.

PB4 is used to input the floppy-disk interrupt (FDIR) status, which has been latched by the floppy-disk processor in the LS259 at B-1 on sheet 4.

PB5 is used to sense the PRES/ reset, derived from the CRES/ bus signal at C-3. The CRES/ signal is pulled low by RESET/ to reset the parallel device. PB5 can be used to reset a parity error on the parallel interface at D-3 on sheet 3.

PB6 is used for the COPS handshake as described above.

PB7 is used to output the CRES/ reset signal to the parallel port interface at pin 117 of the J1 connector.

## 6.7 Miscellaneous Logic

The I/O board contains several blocks of logic, which have been located there for optimal use of board space within the Lisa.

This section describes the hardware implementation of the following functions:

- \* Speaker volume control
- \* Battery power control
- \* Video contrast latch.

### 6.7.1 Speaker Volume Control

The Lisa is equipped with a speaker; its volume can be controlled by the software. The value of the volume is presented by the PB1-PB3 lines of the Keyboard 6522, and these are input to the D-to-A ladder network shown at C-4 on sheet 5.

The resulting analog value is presented as a voltage to pin 10 of device U10A. This operates as a voltage follower to present a high input impedance to the D-to-A ladder network.

The resulting output defines the speaker level, while the TONE output from the 6522 defines the frequency. The output signal is presented to pin 13 of A10, which drives the power transistors Q3 and Q4

to give the speaker output signal.

#### 6.7.2 Battery Supply and Control

The Lisa is equipped with a battery located on the I/O board. It provides a +5 V supply to operate the COPS in the keyboard/mouse controller should the Lisa be completely disconnected from a power source, so that the Time-of-Day clock logic can continue to function.

The battery automatically recharges itself when the Lisa is on. The battery is capable of running the COPS for 10 hours without loss of data. Once battery power is exhausted, the battery supply shuts down to avoid erratic COPS operation.

The circuit is shown at the bottom of sheet 5. It is designed around the 4193 voltage regulator. Recharge power for the batteries is provided by the +12 VDC supply through diode D7. Zener D8 limits the voltage output of the +5 V supply.

The 4193 operates by pulling the LX output to ground, then allowing it to fly back up to produce the output voltage. L1 is the flyback inductor, while capacitor C12 smoothes the output.

The frequency with which the 4193 does this depends on the voltage sensed by the LBR and VF inputs on the precision voltage-divider formed by resistors R19-R21. The VF input looks for a 1.3 V reference. The voltage is higher than this when the +5 VSTBY supply is operational, which supplies a level of approximately 5.7 V. Allowing for a diode drop of 0.3 V across D3 still results in the +5 V level being pulled to over 5 V. At this time, the battery is being recharged.

When the Lisa is disconnected from a power source, the +5 STBY is removed and the +5 V level begins to drop, which triggers the 4193 through the sensing inputs from the ladder to attempt to restore the level. The 4193 is capable of providing an acceptable +5 V output for battery voltages as low as +2.4 V.

The LBR input detects a 4.5 V level on the +5 V output, and pulls the LBD output low. This resets the flop formed by the two gates of CMOS device U14B and pin 4 goes low. This forces Q6 to be back-biased and disconnects the load from the battery output. This is done to avoid cell

reversal, which can destroy nickel/cadmium battery cells.

When the normal 5 V supply is operational, the pin 10 output of the inverter causes the flop to be held off and transistor Q6 is always on.

### 6.7.3 Video Contrast Latch

This is shown in the upper part of sheet 5 of the schematics. The data are input via the parallel port interface 6522A port, which is latched in the CMOS 74C174 at D-3. The data is presented to the summing network, which presents the result of the D-to-A conversion as a DC voltage to pin 3 of the low-input-impedance voltage follower in the first stage of device U10A.

The output of this is fed through R19 to the second stage of device U10A, which functions as an inverting amplifier with a gain of unity. The bias network formed by resistors R10 and R11 provides a bias voltage of approximately 3.5 V to pin 5. The RESET signal pulls this point to ground, which blanks the screen.

The CONT output can vary between +2 V, fully black, and +7 V, fully white. This signal is an input to the video board, which is described in Chapter 7.

## CHAPTER 7. THE VIDEO BOARD

---

The video board provides most of the analog circuitry that operates the Lisa's CRT. The board is mounted on the rear wall of the CRT enclosure and contains hazardous voltages. Extreme care should be taken when using the Lisa with the video board exposed for troubleshooting.

The video digital control logic on the processor board drives the video board. Refer to section 4.5 for a description of the video control.

### 7.1 Block Diagram

To better describe the video board's operation, it has been functionally divided into these major blocks:

- \* Power-supply circuits
- \* Video-amplifier circuits
- \* Vertical deflection circuits
- \* Horizontal deflection circuits.

The video board processes the bit stream of serial display data it receives from the processor board and translates the data into a display by means of the raster scan on the screen. To perform this, signals that synchronize the raster sweep with the data are also provided by the processor board.

An overview of the video board is shown in Figure 7-1 as a block diagram. This chapter refers to the video board schematic, number 050-4012-A, in Appendix D.

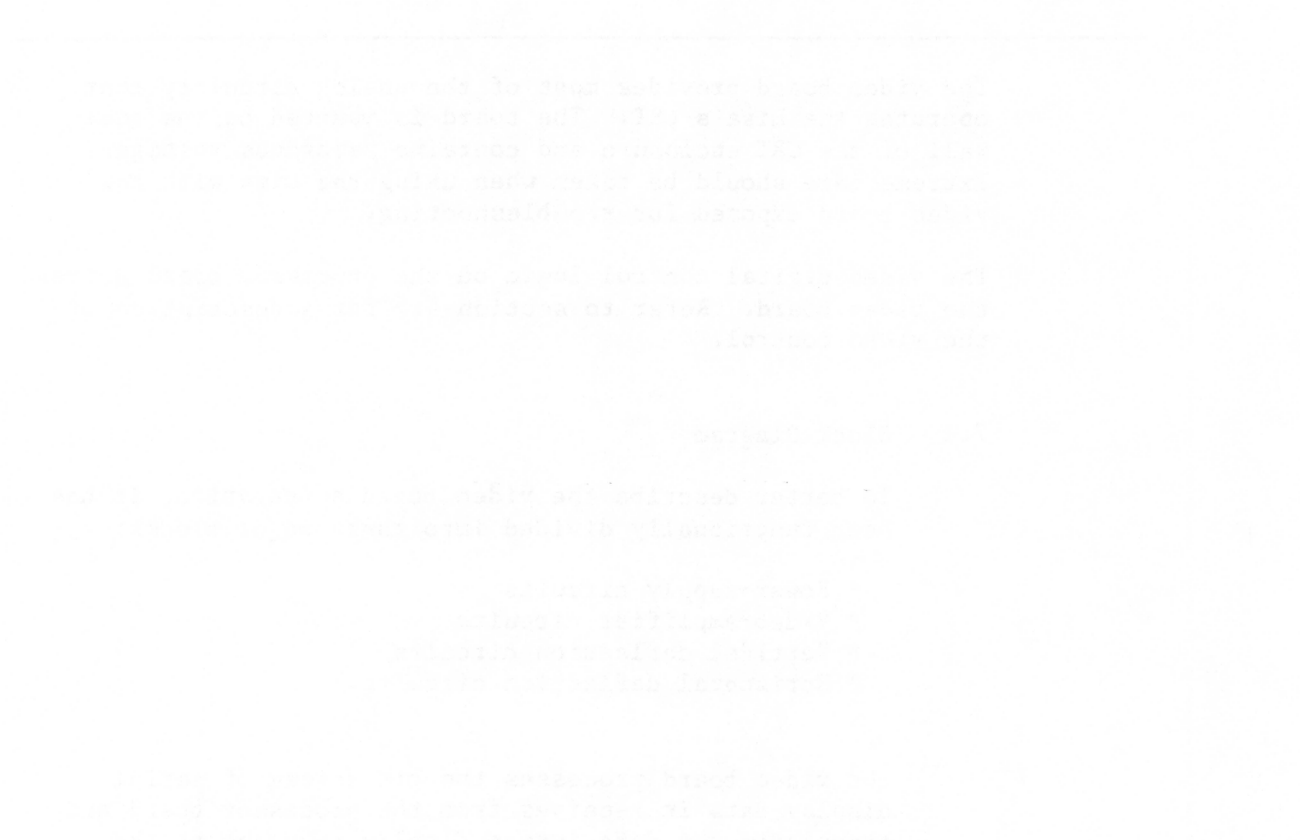


Figure 7-1. Video Board Block Diagram

## 7.2 Power-Supply Circuits

Power is supplied to the video board via the connector at the upper left of the schematic in Appendix D. The +12 VDC is the same as that supplied to other circuits in the Lisa. The +33 VDC is used exclusively by the video board. Refer to Chapter 10 for a discussion of the power supply and its capabilities.

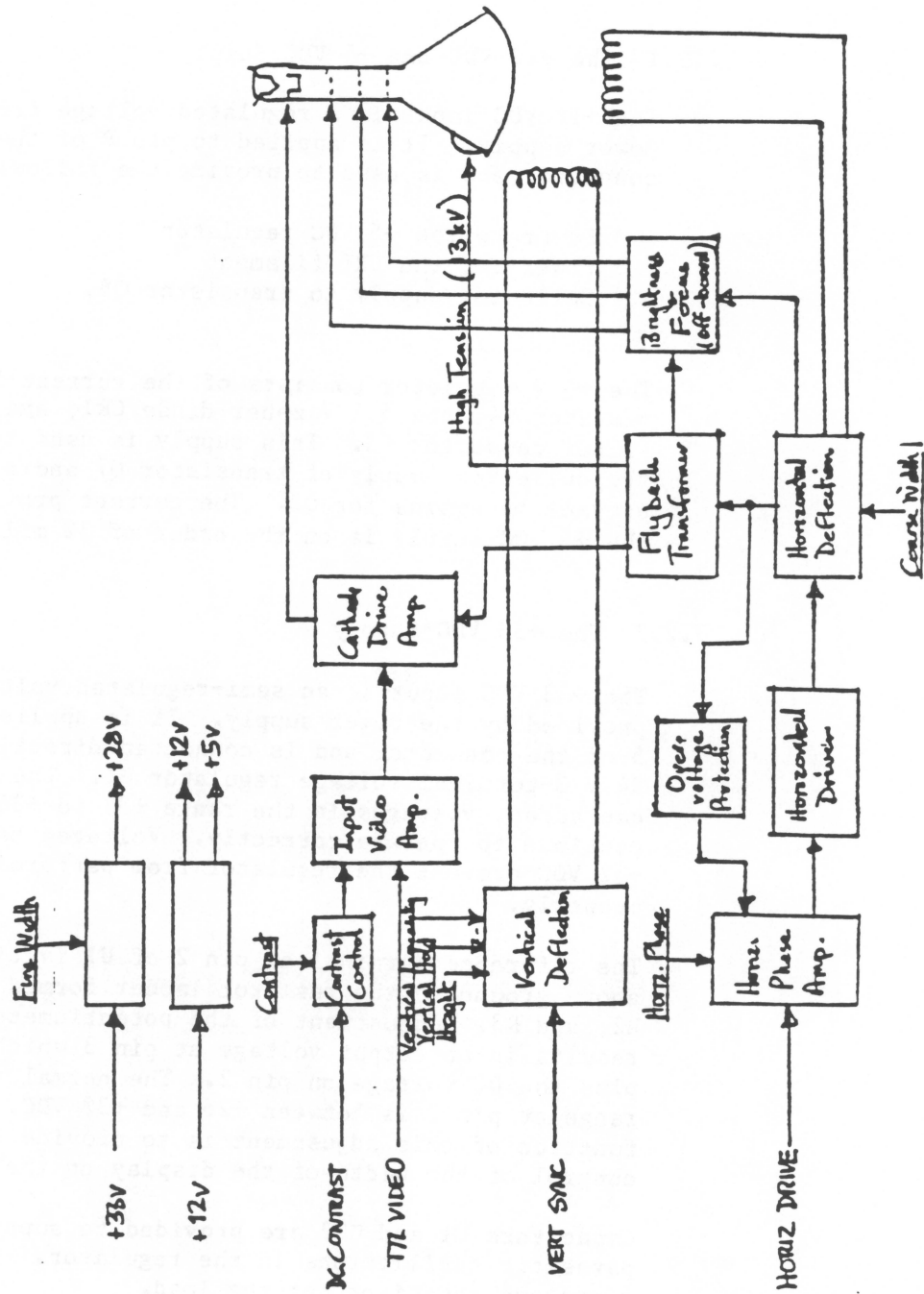


Figure 7-1. Video Board Block Diagram

### 7.2.1 The +12 VDC and +5 VDC Supply

The +12 VDC input is a regulated voltage from the power supply. It is applied to pin P of the connector and is used to provide the following:

- \* Power to the +5 VDC regulator
- \* Power for the CRT filament
- \* Collector supply to transistor Q8.

The +5 V regulator consists of the current-limiting resistor R4, the 5.1 V zener diode CR1, and the filter capacitor C3. This supply is used to provide the collector supply of transistor Q7 and also to provide base-bias for Q2. The current provided by the +5 VDC supply is on the order of 30 milliamps.

### 7.2.2 The +33 VDC Supply

The +33 VDC input is an semi-regulated voltage provided by the power supply. It is applied to pin 5 of the connector and is connected directly to the 24 V 3-terminal voltage regulator U1. The regulator can accept voltages in the range +33 to +36 VDC and continue to operate correctly. Voltages below about +32 VDC prevent the regulator from performing properly.

The reference terminal at pin 2 of U1 is biased above ground by the resistor ladder formed by R1, R2, and R3. Adjustment of the potentiometer R2 results in an output voltage at pin 3 which is 24 V plus the DC voltage on pin 2. The normal voltage range at pin 3 is between +26 and +30 VDC. The function of this adjustment is to provide a fine control of the width of the display on the CRT.

Capacitors C1 and C32 are provided to suppress parasitic oscillations in the regulator. C2 acts as a storage capacitor for the load.

The regulated +28 VDC, nominal, provided by the supply provides power for the rest of the video board and the CRT.

## 7.3 The Video-Amplifier Circuit

The function of the video-amplifier circuit is to take the serial TTL data provided by the processor board and convert the digital information into black, logical 0, or white, logical 1, pixels on the screen by controlling



the cathode emission in the CRT.

### 7.3.1 Video-Data Input and Contrast Control

TTL-level video data are presented to the video board on pin 1 of the connector. It is impressed on the base of transistor Q6 by way of the current-limiting resistor R51. Resistors R51 and R52 form a voltage-divider network to increase the noise-rejection of Q6, which acts as a high-speed switch. The collector of Q6 is therefore at ground or at a DC level, depending on the digital state of the TTL input.

The actual DC voltage level is controlled by the contrast control circuitry. The programmed contrast DC level is provided by circuitry on the I/O board. Refer to Chapter 6 for details of this signal. Potentiometer R5 provides manual adjustment of the level actually presented to the emitter-follower amplifier formed by transistor Q1. Resistor R6 and capacitor C4 make up a low-pass filter that provides smoothing of the incoming DC contrast level.

Transistor Q1 is connected as an emitter-follower in order to provide the high current required to drive the low impedance of R8. The higher the voltage on R8, the larger the contrast between black and white in the final pixel on the CRT.

Note that there are no gray levels in the Lisa video circuitry. Gray tones can be generated by pixel interleaving under software control. The details of this lie outside the scope of this manual.

### 7.3.2 Cathode Drive Circuit

The collector output of transistor Q6 is passed on to the base of the stacked transistor pair Q2/Q3. The advantage of this circuit is that it combines the advantages of the high gain of Q3 with the ability to withstand high voltages of Q2. Capacitor C6 and resistor R11 are used for high-frequency peaking. Resistor R10 prevents transistor Q2 from going into cutoff when the screen is blank.

The collector load for Q2 consists of the peaking coil L1 and collector load resistor R14, with R13 providing a shunt to damp L1. The collector voltage on Q2 is dependent on the polarity of the pixel being displayed. For black, it is about +60 VDC, while for white it is from 10 to 30 V lower. The



exact voltage level for white depends on the setting of the contrast R5 and the input DC contrast level at pin B on the board control circuit.

The flyback transformer assembly at B-2 of the schematic provides a +60 VDC power source at pin 2 on connector P-3. This is filtered by the storage capacitor C8. Capacitor C7 and diode CR3 form a decoupling network, which ensures that the beam current to the cathode is removed when the Lisa is turned off. This is known as a "spot-killer" and prevents phosphor burn at one point on the screen.

The output of Q2 is passed through the 220 ohm arc-protecting resistor in the cathode lead. This prevents any arcing voltages in the tube from reaching the video amplifier circuitry. Diode CR2 is normally reverse-biased. In the case of an arc occurring in the CRT, this shunts the high-voltage on the lead, acting as an arc-suppressor. The lead connects to pin 2, cathode, of the CRT itself.

#### 7.4 Vertical Deflection Circuit

The vertical deflection circuit positions the horizontal scan of the CRT at the appropriate position in the vertical axis. It is shown in the left center part of the schematic, and consists largely of integrated circuit U2 and its associated components.

Vertical synchronization pulses are provided by the processor board on pin A of the connector. This TTL signal is passed through coupling capacitor C9 and current-limiting resistor R17. The resulting signal pulses are fed into the input of the TDA 1170 at pin 8.

##### 7.4.1 Vertical Deflection Oscillator

The resistor/capacitor (RC) network formed by C10 and R19/R20 provides a timing period for the oscillator internal to U2. Potentiometer R19 enables the period to be varied into synchronization with the periodic screen refresh. This acts as a vertical hold. The network connects to pins 6 and 9 of U2.

##### 7.4.2 Vertical Voltage Amplifier

Internal to U2 is a circuit used to control the amplitude of the vertical deflection.

The gain of this amplifier is controlled by the amount of current sunk to ground from pin 7 of U2. This is a function of the resistance provided by R21 and R22. Potentiometer R22 allows this value to be varied; this adjusts the amplitude of the vertical saw-tooth voltage and thus the physical height of the display on the CRT.

The voltage amplifier output is present on pin 1 of U2. It is utilized in two distinct places:

1. As an input to the feedback network formed by R31, R32, C15, and C16. This controls the linearity of the sawtooth waveform that is output at pin 1. This can be adjusted by potentiometer R31. The network feeds back to pin 12 of U2.
2. As an input to the power amplifier stage through summing resistor R29 to pin 10 of U2.

#### 7.4.3 Vertical Power Amplifier

Input to the power amplifier is at pin 10. The amplified output is available at pin 4 of U2. The output signal follows four paths:

1. It is fed back into the power amplifier via C14 and R27 to limit the high-frequency response and to prevent parasitic oscillations.
2. It is passed through the vertical deflection coils via pins 1 and 2 of connector P1 and is passed to ground via resistor R33 and capacitor C19.
3. It passes through the snubbing network R25 and R17 to ground. The circuit is in shunt with the deflector coils and prevents any ringing effects in the coils themselves.
4. It takes the signal DC component, sampled by R26 and filtered by C18, and passes it on to the summing resistor R28, and so to the amplifier input signal on pin 10. This signal is used to establish the DC operating point of the power amplifier to ensure DC stability.

AC current passing through the deflection coils is

sampled across R33, fed through summing resistor R30, and also presented as a component of the input signal on pin 10. This controls the AC voltage gain of the power amplifier and enhances AC stability.

#### 7.4.4 Bootstrap Circuit

The power input to U2 is pin 2, which should have a DC voltage between +11 and +16 VDC. This is provided by the voltage drop from +28 V, which in turn is provided by R24 with storage capacitor C12. This voltage is also used to provide the collector supply for the emitter-follower circuit of Q1 in the video-amplifier section.

Part of the output signal of the power amplifier, presented at pin 4 of U2, is also presented at pin 5. This forces pin 5 above the DC input voltage at pin 2 and reverse-biases CR5.

This forces pin 3 above the DC input voltage because of the charge stored in C11.

### 7.5 Horizontal Deflection Circuits

The horizontal deflection circuits provide the timing and control of the sweep of the beam across the CRT in the horizontal direction.

The horizontal frequency of the Lisa video circuit is approximately 22.7 KHz.

#### 7.5.1 Horizontal Input Circuit

The horizontal drive pulse is a TTL signal provided by the processor board and presented to the video board on pin C of the connector.

Summing resistor R53 presents the signal to Q7, which operates as an adjusted Miller integrator. Adjustment is achieved by controlling the amount of feedback to the input by way of the horizontal phase-control potentiometer R34. Feedback to the base of Q7 is by way of resistor R36 and the Miller capacitor C20.

This circuit controls the amount of rise and fall time in the output signal by means of the Miller effect. This controls the switching point of Q8 in time, which operates as a high-speed saturated switch.

### 7.5.2 Horizontal Sweep Amplifier

The output of Q8 is used to drive the base of the horizontal driver transistor Q4. The output of Q4 is in turn coupled to the base of the horizontal output transistor Q5 by way of the coupling transformer T1 and the base-current limiting resistor R45.

The RC network formed by R41 and C24 is used to control the damping across the primary windings of T1. Similarly, R44 is used to damp the secondary windings.

Power for the driver stage of the amplifier is derived from the +28 VDC supply via the decoupling network formed by R39 and C23.

### 7.5.3 Horizontal Deflection

Transistor Q5 turns off when the base voltage goes negative. When this occurs, the stored current in the horizontal flyback transformer and the horizontal deflection coils collapses. This causes the collector voltage of Q5 to rise to approximately 280 V. The rate of rise is controlled by the timing capacitor C25, which lies between the collector of Q5 and ground. Diode CR9 is used as a damper to prevent the collector of Q5 from going negative by passing the negative half of the current cycle to ground.

Energy is coupled out of the horizontal circuit by way of the flat-faced correction capacitor C26. Current is then passed through the horizontal deflection coils, through the horizontal linearity coil L2, and to ground. Coarse picture width is controlled by varying the inductance of L3, which is in parallel with the deflection coils.

Coil L2 increases the linear current ramp through the deflection coils. A magnet in the coil is used to bias the coil-1 in one direction. As current is passed, this adds to or subtracts from the magnetic bias. This in turn alters the value of the inductor. R46 is used to damp any high-frequency oscillation within the horizontal deflection coil circuit.

#### 7.5.4 Horizontal Flyback

The flyback transformer also connects to the horizontal output transistor Q5. It is used to step the voltage on the collector of Q5 either up or down.

This is used to generate the high-voltage supply for the CRT at 13K VDC, the 60 VDC supply for the video amplifier and also +600 VDC for the G2 voltage on pin 6 in the CRT. It also provides power for the brightness and focus circuit, which is physically located on the power supply board in order that the controls are available to the operator of the Lisa.

Power input to the flyback assembly comes from the +28 VDC supply on pin 3, violet.

A portion of the horizontal output signal is coupled by C27 to R47 and CR10. This provides a -100 VDC source to operate the brightness and focus circuits.

#### 7.5.5 Overvoltage Crowbar

The voltage at the collector of Q5 is sampled by the precision divider formed by R42 and R43. This is fed through the peak-detector circuit formed by CR8 and C22 and placed on the cathode of CR7. If the voltage on the collector of Q5 should rise above acceptable limits, CR7 conducts and provides a forward bias on SCR CR6. This latches CR6 on, shunting the collector supply of Q8 to ground. It also turns the horizontal circuits off and avoids a damaging high voltage on Q5. SCR CR6 remains in this condition until power is removed from the Lisa. It also prevents excessive voltage to the CRT, thus keeping it within the safe X-ray rating of the tube.

#### 7.5.6 Brightness and Focus

The control circuits for these two functions are logically a part of the horizontal deflection circuit, but are physically a part of the Lisa power supply for ease of operator access to these controls.

The +600 VDC high-voltage power for these circuits is provided by the components CR11, R48, and C33 from the secondary windings of the flyback transformer. The negative voltage comes from

components C27, R47, and CR10.

The brightness of the CRT is controlled by adjustment of the DC voltage on pin 1, the control grid, of the CRT. This can vary from +12 to -38 VDC with respect to ground, and is adjusted by means of the 2 V megohm resistor marked Brightness.

The focus is a voltage placed on pin 7 of the CRT. It can vary between -100 and +300 VDC, depending on the adjustment of the 2 megohm resistor marked Focus.

Reference is made to the fact that the

The purpose of the fact is to show that the  
the fact is to show that the fact is to show that the  
the fact is to show that the fact is to show that the  
the fact is to show that the fact is to show that the

The fact is to show that the fact is to show that the  
the fact is to show that the fact is to show that the  
the fact is to show that the fact is to show that the  
the fact is to show that the fact is to show that the



## CHAPTER 8. USER INTERFACES

---

The Lisa uses the CRT and a speaker to communicate with the user. The user can communicate with the Lisa by using the mouse and keyboard. In addition, there is a video jack available on the motherboard at the back of the Lisa that supplies a composite video signal and can be used to drive a secondary CRT.

### 8.1 The CRT

The CRT is controlled by a combination of Lisa software and hardware elements on both the processor and video boards. Refer to section 4.5 and to Chapter 7 for a discussion of the hardware.

### 8.2 The Keyboard

The 76-key Lisa keyboard is a detachable assembly that contains a full key-set and function keys. It is connected to the Lisa by a standard 1/4 inch stereo phone jack, located above the on-off button on the front of the cabinet. The standard keyboard layout, North American, is shown in Figure 8-1. Appendix H contains all available Lisa keyboard layouts.

#### 8.2.1 Keyboard Logic

The keyboard logic is shown in schematic 050-4001 in Appendix E. It operates under control of the COPS device shown on the schematic at C-1.

The COPS is identical to that found in the keyboard control logic on the I/O board. Refer to section 6.5 for details of this. The same routines are present in both devices. Only the routines that apply to the location of the device are used.

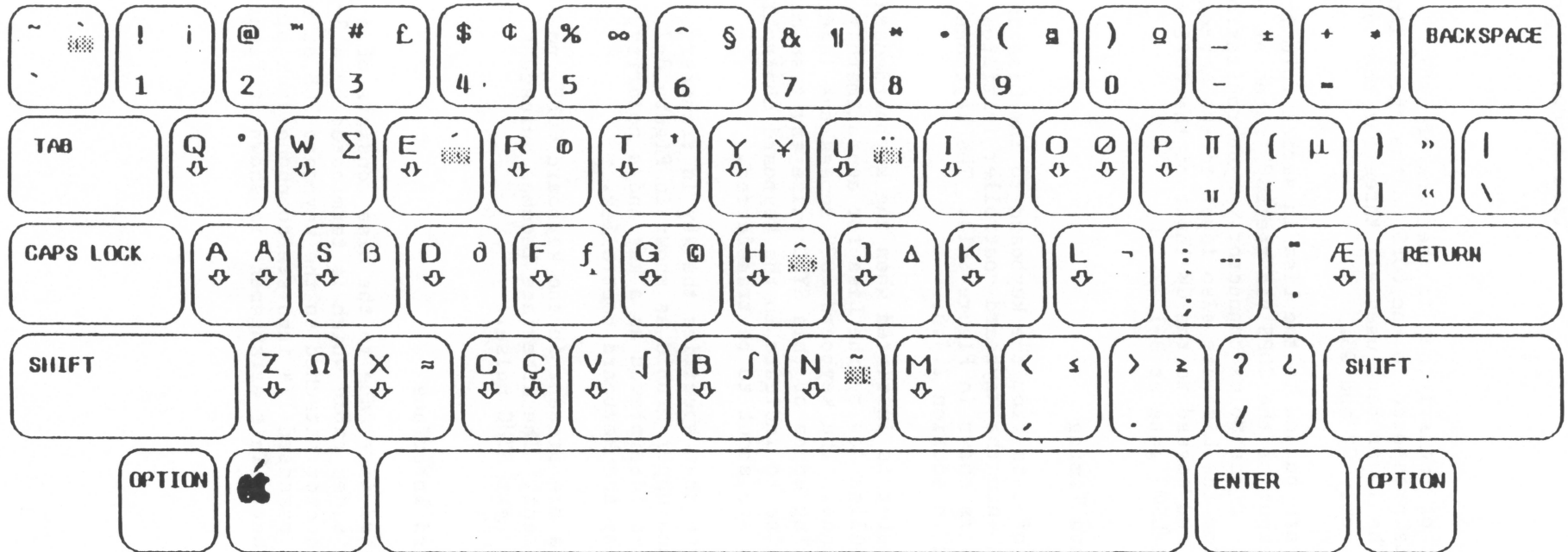


Figure 8-1. Standard Keyboard Layout

The keys interface to five 4067 16-to-1 multiplexer devices. The COPS polls these in an upper and lower bank by means of the SK signal and the LS03 gate at C-2. The configuration of the COPS D0-D3 outputs defines which of the sixteen switches attached to each device is being selected.

## US NEW Keyboard Layout

TN / 8 Nov, 82



## LEGEND



Left



**Right**



Up



Down

Dead key, no character is generated until next key is pressed.

⬇ Character has lower case  
and is affected by Caps Lock.

### OPTION Selects Alternate Keyboard

Left:	Right:
Primary keyboard	Alternate keyboard



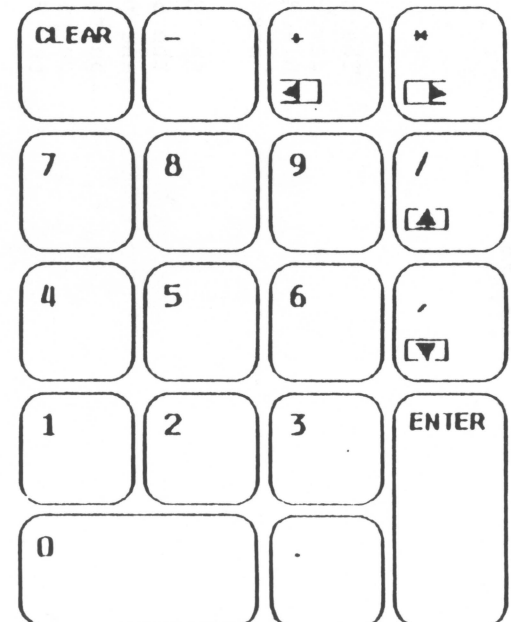
Upper: Shifted

Lower: Unshifted

**Boldface** indicates that the character is printed on the keytop.

When no character is shown in the lower area of the keytop, one of the following is true:

- If the symbol  $\Downarrow$  is shown, the lower character is the lower case equivalent of the upper character.
- If the symbol  $\Downarrow$  is not shown, the lower character is the same as the upper character.



The state of the switches in the bank of 4067 devices is presented to the COPS on the three lines G1-G3. The lower bank uses all three but the upper bank uses only G2 and G3.

Key data are passed to the Lisa in serial form from the S0 output of the COPS via the LS03 gates at B-1 and pin 1 of the Molex connector. Synchronization pulses from the Lisa are also input to the keyboard on pin 1 and passed to the G0 input of the COPS via the final LS03 gate at B-1.

### 8.2.2 Keyboard Timing

Transfer of data from the keyboard to the Lisa is performed when the keyboard controller initiates it. This is shown in Figure 6-13. The keycodes are discussed in section 2.4.5.

Data transfer is initiated when the keyboard-data line is pulled low by the Lisa for approximately 20 microseconds. The keyboard COPS senses the leading and trailing edges of this SYNC pulse and transmits an ACK pulse to indicate to the keyboard controller that data are about to be transferred.

A data byte that specifies the key is transferred in a lower and upper nibble as shown in Figure 8-2. The byte is interpreted as a key and a polarity, up or down, by the keyboard controller.

If no data are present in the keyboard COPS, no ACK pulse is sent. The interface becomes quiescent until the next SYNC pulse.

### 8.2.3 Keyboard Interface

The keyboard interface to the Lisa consists of a 3-wire shielded cable which is terminated by a 3-pin Molex connector attached to the keyboard PCB at one end and a standard 1/4 inch stereo phone jack plug at the other. This arrangement is shown in Figure 8-2.



Figure 8-2. Keyboard Interface

### 8.3 The Mouse

The mouse is an electromechanical device that provides communication with the Lisa software in addition to the keyboard. It consists of a rolling ball arrangement on the under side and a plastic cover with a button on the top. A cable connects the mouse with the 9-pin DB connector in the center of the motherboard at the back of the Lisa.

The specific functions of the mouse depend on the software currently running. However, the two controls on the mouse provide the same general function in all cases:

- \* Rolling the mouse around a flat surface moves a cursor around the screen, and
- \* Pressing the button on top of the mouse selects the item or software function at the cursor location.

#### 8.3.1 Internal Components

The internal components of the mouse consist basically of a switch and two directional wheels. One directional wheel detects motion forward and

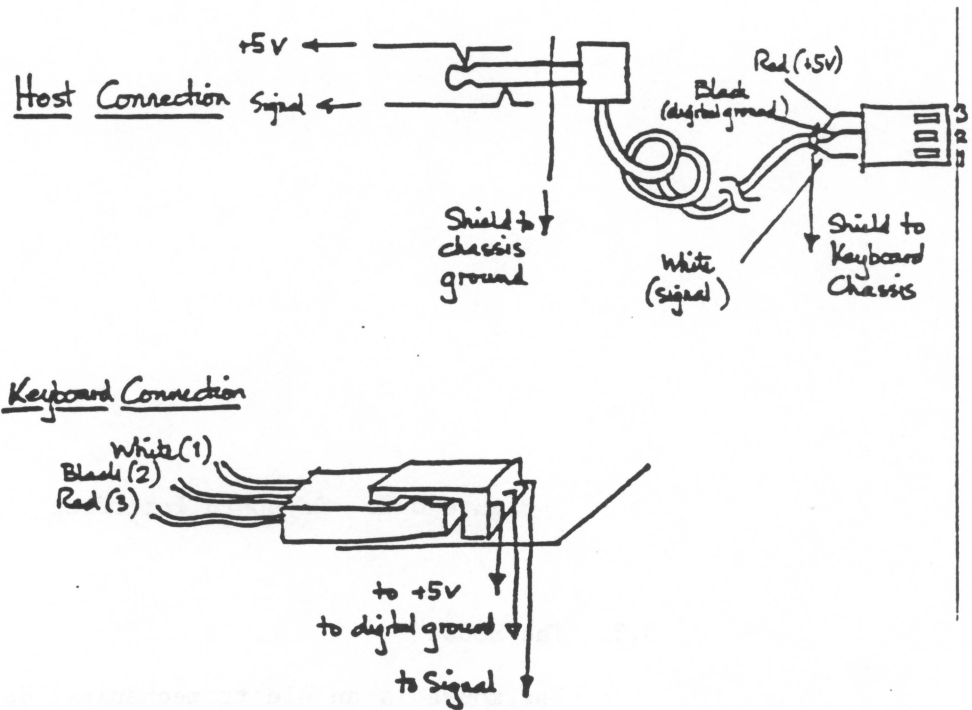


Figure 8-2. Keyboard Interface

back, corresponding to up and down on the screen, while the other detects motion left and right.

Each wheel has a number of slots cut around the rim. These pass two photoelectric diode assemblies as the mouse is moved in that axis and the wheel turns. The two diode assemblies produce a series of pulses to the mouse interface on the I/O board.

The relationship of one set of pulses to the other indicates the polarity of the direction in which the mouse is moving, and the number of pulses is proportional to the distance travelled. Refer to Figure 8-3, which illustrates this.

### 8.3.2 The Mouse Interface

The mouse interface is briefly described in section 3.5. The 9-pin DB connector is located in the middle of the connector panel at the back of the Lisa. Refer to Figure 3-11 for the connector layout.

Only SWITCH0 is operational in the mouse. Each pair of the LEFT/RIGHT and UP/DOWN inputs pulses, as shown in Figure 8-3, when the mouse moves one increment in that direction. One increment equals a constant unit of movement in that direction on the flat surface.

Figure 8-3. Mouse Movement Waveforms

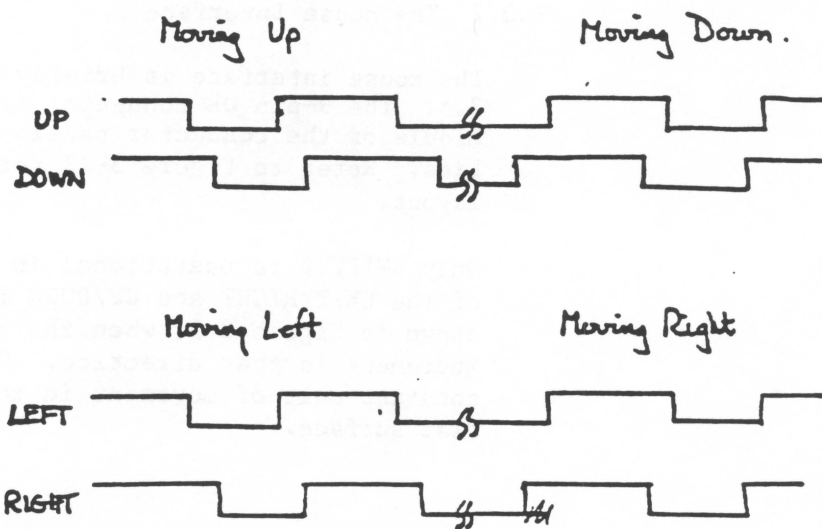


Figure 8-3. Mouse Movement Waveforms



## 8.4 Other User Controls

Several user-oriented features of the Lisa are described elsewhere in this manual in more detail but are mentioned here for convenience.

### 8.4.1 The On-Off Switch

The on-off switch is located on the front of the cabinet, in the lower-right corner. Pressing this switch when the Lisa is off sends power to all parts of the Lisa under control of the boot software.

Pressing this switch when the Lisa is on causes an interrupt to software which in turn cleans up its environments and then ejects any floppy disks in use. The Lisa is then turned off under software control. This turns off everything except the control COPS, which handles the RTC and power-on control. The control COPS receives its power from a standby supply in the power supply. Refer to section 10.5 for details.

Whenever the Lisa is isolated from a power source, perhaps because the Lisa is disconnected or the power fails, a self-recharging battery maintains the voltages necessary to operate the standby supply voltages for a period up to approximately 10 hours. The battery is automatically recharged whenever the Lisa is turned on. Refer to section 6.7 for a discussion of this.

### 8.4.2 The RESET Switch

The RESET switch is located on the motherboard at the back of the Lisa, next to the video connector. RESET is used to restart the Lisa in the event that an unrecoverable error has put the Lisa in an undefined state. This should never be activated during normal processing or loss of data almost certainly results. The reset switch is shown at location C-2 on the motherboard logic schematic 050-4013 in Appendix G.

### 8.4.3 The Speaker

The built-in speaker can be used to provide an audible warning to the user. It is controlled by logic on the I/O board under software control for

volume and tone. Refer to section 6.7 for a discussion of speaker control.

#### 8.4.4 The Composite Video Output

The composite video output is available on the J10 video jack at the back of the motherboard. The circuit is located on the motherboard and uses the CVOUT and VID signal outputs from the processor board. Refer to section 4.5. The composite video signal is output to the jack via resistor R2.

The logic is shown on schematic 050-4013 in area C-2. Both the VID and CVOUT signals are effectively ORed to the base of transistor Q2. This controls the amount of current passing through Q2 and consequently through R7, which controls the amount of current passing through Q1.

## CHAPTER 9. FLOPPY-DISK DRIVES

---

The Lisa is equipped with two floppy-disk drives for data file storage. These are located one above the other at the right side of the cabinet. They interface to the Lisa by means of a floppy-disk controller, which is located on the I/O board and is described in section 6.2.

This chapter does not present an exhaustive description of the drive. Refer to other documentation on the disk drive for further details.

### 9.1 Specifications

Capsule summaries of disk-drive specifications follow.

#### 9.1.1 Media

Disk size . . . . .	5 1/4 inch
Media . . . . .	qualified double-sided
	10,000 FCPI
Track density . . . . .	62.5 TPI

#### 9.1.2 Speed

Motor speed settle time . . . . .	150 msec
Motor on time . . . . .	400 msec
Head settling time . . . . .	10 msec
Carriage movement . . . . .	linear stepper actuator
Spindle rotation . . . . .	DC Motor

#### 9.1.3 Electrical

+5 VDC standby. . . . .	75 mA
+5 VDC running . . . . .	80 mA
+12 VDC standby . . . . .	42 mA
+12 VDC running . . . . .	850 mA
+12 VDC maximum (during motor start) . . . . .	1.6 A
-5 VDC standby . . . . .	5 mA
-5 VDC running . . . . .	8 mA

#### 9.1.4 Environmental

See section 3.2, Lisa Specifications.

### 9.2 Drive Block Diagram

The floppy-disk drive is an assembly that enables 5-1/4 inch flexible diskettes to be used as data storage media. In the Lisa, there are two functionally identical drives controlled by logic on the I/O board.

The drive itself can be thought of as consisting of a number of component blocks, as shown in Figure 9-1. The motors and detectors present in the drive chassis are directly controlled by the digital board, while the head read/write functions are performed by the analog board under control of the digital board.

Figure 9-1. Drive Block Diagram

### 9.3 Drive Interface

The interface to the drives is performed in the Lisa by signals provided on motherboard connector J1, under control of the I/O board's floppy-disk controller. Refer to section 6.2.

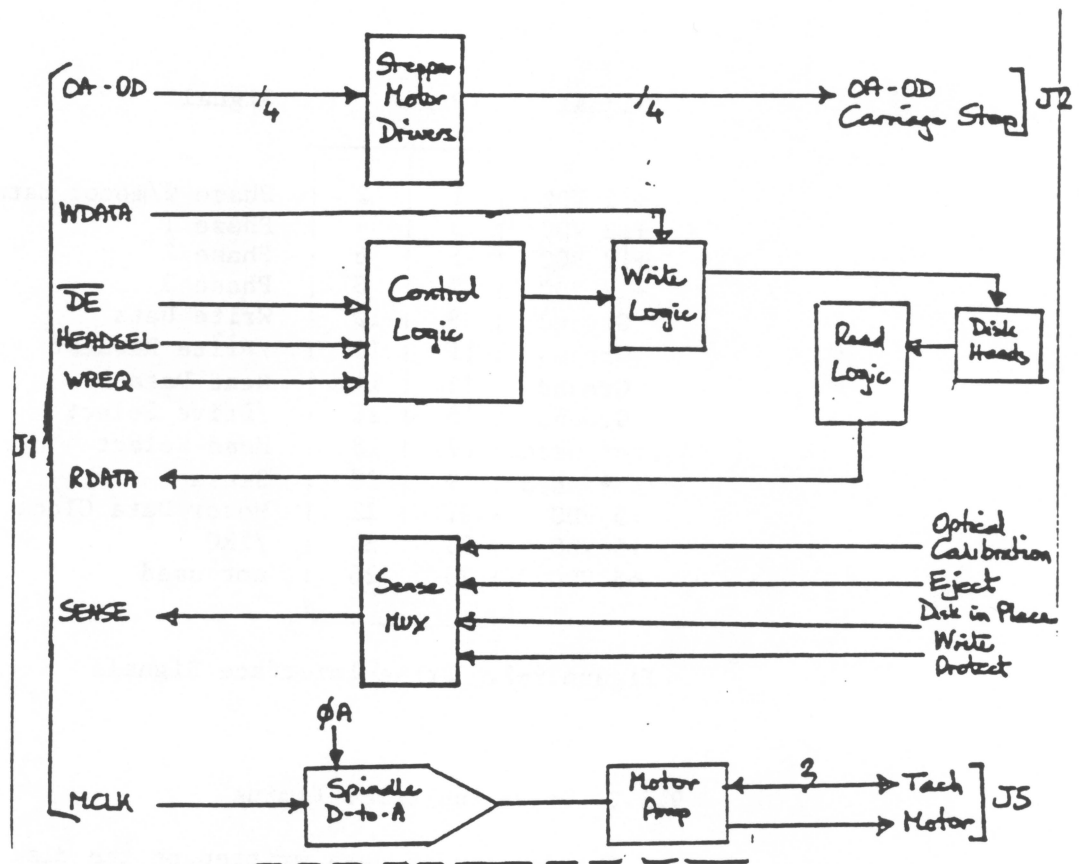


Figure 9-1. Drive Block Diagram

### 9.3.1 Drive-Interface Signals

The interface signals to the drive are shown in Figure 9-2.

Signal	Pin #		Signal
+12 VDC	1	2	Phase 0/motor data/sense select
+12 VDC	3	4	Phase 1
+12 VDC	5	6	Phase 2
+12 VDC	7	8	Phase 3
Ground	9	10	Write Data
Ground	11	12	/Write Request
Ground	13	14	Read Data
Ground	15	16	/Drive Select
not used	17	18	Head Select
not used	19	20	Sense
+5 VDC	21	22	Motor Data Clock
+5 VDC	23	24	/IRQ
+5 VDC	25	26	not used

Figure 9-2. Drive-Interface Signals

### 9.3.2 Drive-Interface Timing

The density of data written on the disk is a function of the floppy-disk controller and the capacity of the medium to store bit transitions. Refer to sections 6.2 and 6.3 for a discussion of controller operation.

Timing of the stepper motor that controls head movement is defined by the four phase lines on the interface. These are manipulated to move the heads forward or backwards.

In order to avoid inaccurate head positioning due to mechanical tolerances in the carriage mechanism, all seeks finish with a final motion in a direction towards the optical calibration point. Note that this means that the upper head is moving away from the spindle while the lower is moving towards it. Refer to Figure 6-11.

In order to overcome the inertia during carriage

acceleration and deceleration, three different periods are employed between speed transitions on the phi lines. This can be seen in Figure 9-3.

Figure 9-3. Carriage Movement Timing

In the quiescent state, all four phase lines are low. Motion is begun by asserting two of the lines. In the forward direction, towards the optical calibration point, this would mean phi3 and phi0; in the reverse direction, phi0 and phi1. At the end of the first period, one phase is deasserted and another asserted, as shown in Figure 9-3. Note that one phase remains on while others are being changed.

When the carriage is at the required position, phi0 is always the last phase to remain asserted alone for the final period before all phases are again deasserted and the carriage is at rest.

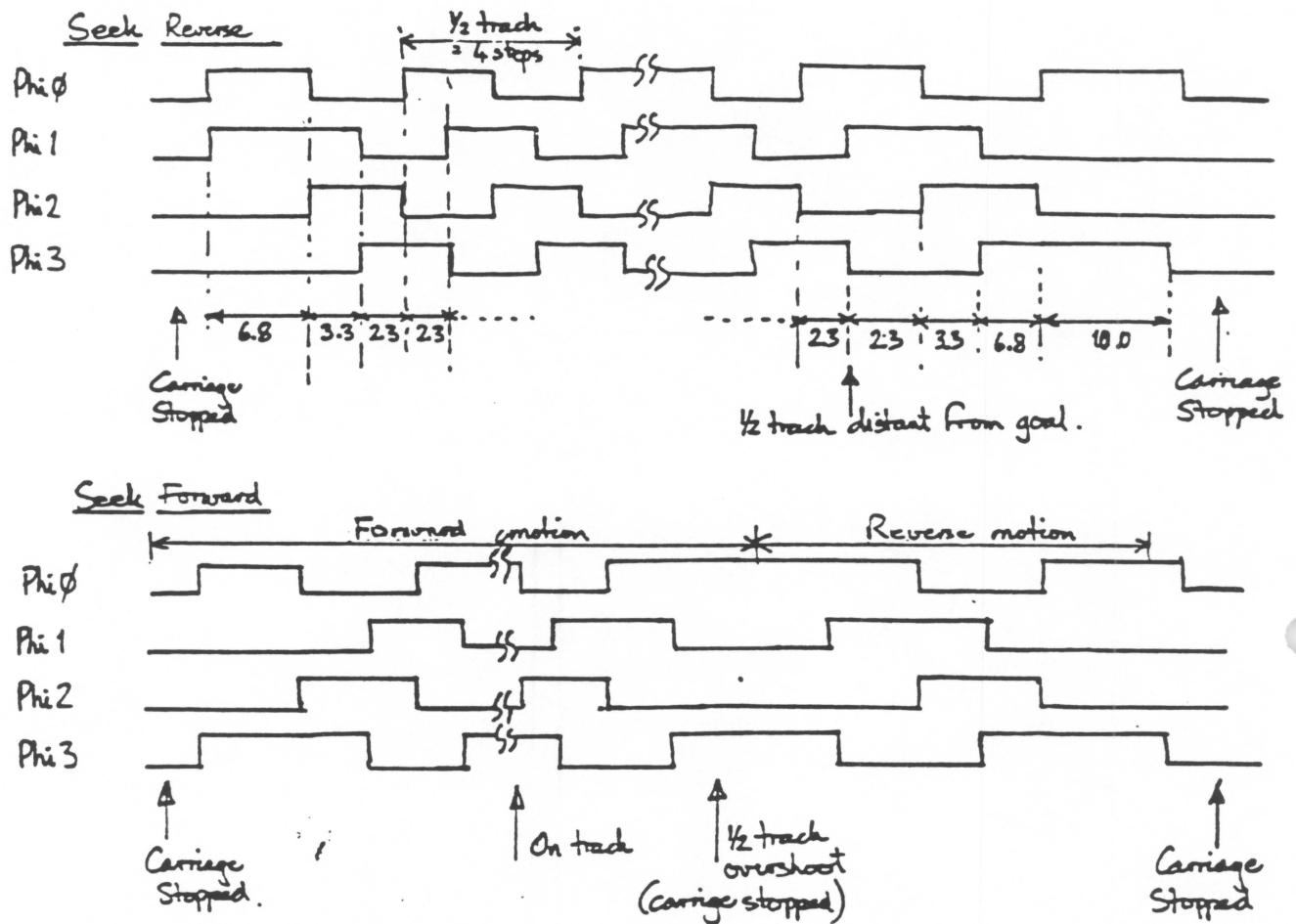


Figure 9-3. Carriage Movement Timing



Note that for a seek in the reverse direction, the carriage is brought to a halt four steps beyond the desired position, and then a seek forward of four steps is made to restore the head to the center-track position. This operation is known as overshoot.

Each step corresponds to 1/8th of a track. Thus all tracks are eight steps apart. In the case of difficulty in reading data from the required track, the floppy-disk controller attempts to offset by 1/8th track steps to allow for misalignment.

The optical disk calibration signal is available to the controller as an aid in overcoming differences in mechanical alignment between disks. It provides an accurate radial reference point from which seeks can be made. This is also known as the recalibrate function.

#### 9.4 Basic Drive Operation

The drive is capable of performing a number of operations under control of the signals on the drive interface. The operations are described in following subsections.

##### 9.4.1 Inserting and Removing Disks

When a floppy disk is inserted in the drive, the eject mechanism is armed and a switch is activated to indicate that the disk is in place. This can be detected by the controller on the STATUS signal line, pin 20, if both the HEADSEL, pin 18, and the phi0, pin 2, are asserted by the controller.

Pressing the eject button on the front of the drive causes a line to be asserted in the drive. This state can be detected on the STATUS line by holding the HEADSEL signal asserted with the phi0 signal deasserted.

The controller should then move the heads away from the disk until the disk-in-place signal becomes deasserted. At this point, the heads should be moved nine tracks further in order to activate and reset the eject mechanism.

#### 9.4.2 Loading Disk Heads

When a disk is detected in the drive, the disk heads can be loaded. This is performed in the sequence below. Namely, the:

- \* Drive motor is activated and brought up to speed,
- \* Centering cone is pressed into the disk hole,
- \* Spindle begins rotating,
- \* Disk is clamped to the spindle,
- \* Heads are moved inward towards track 45,
- \* Head load mechanism presses the heads against the disk surfaces.

The steps in the above sequence are performed mechanically when the carriage is being moved to bring the heads over the disk surface. The spindle begins to rotate when clock pulses are presented to the spindle motor control via the MCLK line on pin 22.

The spindle is already spinning before being brought into contact with the disk. The clamber forces the disk to be seated centrally on the spindle with the rotation of the spindle assisting in this.

The heads are moved by pulsing, in the correct sequence, the four control lines of the stepper motor:

- \* phi0, pin 2
- \* phi1, pin 4
- \* phi2, pin 6
- \* phi3, pin 8.

#### 9.4.3 Positioning Heads

The two read/write heads in the drive are mounted on a single carriage that is driven by a head positioning actuator. This consists of a linear stepper motor and lead screw.

This mechanism not only provides head-to-track positioning but also actuates a disk clamping and unclamping mechanism. The mechanism clamps the disk

as the heads are moved into the reading surface and unclamps it as the heads are moved clear of the surface.

#### 9.4.4 Data Read/Write

The ceramic heads in the drive operate in direct contact with the disk surfaces. Each head is a single element with straddle erase elements to clear the inter-track gap.

During a read operation, the heads transform the polarity reversals on the disk surface into a stream of analog pulses. These are converted to serial digital data and presented to the controller as the RDATA signal on pin 14. It is the function of the controller to interpret this data.

The write circuitry takes the serial digital data presented on the WDATA line, pin 10, by the controller and converts it to the correct analog pulses for the write heads.

Data are written to whichever head is currently selected, depending on the write protection condition of the disk currently in place. The controller senses the write protection condition of a disk by deasserting both the HEADSEL and A0 signals before polling the SENSE line.

The write-protect sense condition informs the controller that the disk currently inserted has its write-protect slot uncovered. There is a hardware protection circuit in the drive to inhibit writing to a disk that is write-protected. It is a function of the controller to decide whether or not the disk will be written to. Currently, the floppy-disk controller does not write to a protected disk.

The first part of the report is a summary of the work done during the last year. It is followed by a detailed account of the work done during the last year.

### Summary of work done during the last year

The first part of the report is a summary of the work done during the last year. It is followed by a detailed account of the work done during the last year.

The first part of the report is a summary of the work done during the last year. It is followed by a detailed account of the work done during the last year.

The first part of the report is a summary of the work done during the last year. It is followed by a detailed account of the work done during the last year.

The first part of the report is a summary of the work done during the last year. It is followed by a detailed account of the work done during the last year.

The first part of the report is a summary of the work done during the last year. It is followed by a detailed account of the work done during the last year.

## CHAPTER 10. POWER SUPPLY

The power supply is a flyback-type switcher supply. The circuit diagram is shown in schematic 050-4011, which can be found in Appendix F. The supply provides the power levels and voltages shown in Figure 10-1.

Voltage	Variance	Current
+ 5 V	+/-5% (5.25 V/4.76 V)	4.00 to 8.00 A
+12 V	+/-8% (13.0 V/11.1 V)	0.35 to 2.00 A
+33 V	+36 V Max. +32 V Min.	0.30 to 0.70 A
- 5 V	+/-10% (-5.5 V/-4.5 V)	0.00 to 0.20 A
-12 V	+/-10% (-13.2 V/11.8 V)	0.01 to 0.20 A
+5 VSTBY	+/-5% (5.25 V/4.75 V)	0.00 to 0.10 A

NOTE: The standby +5 V is not switched off by the on-off signal.

Figure 10-1. Supply Current Output

In addition to the main power supply, an auxiliary standby supply is required. The standby supply provides the minimal +5 V power required to run the soft-power-on and real-time clock circuitry while the Lisa is in a power-off condition.

The focus and brightness controls for the CRT are located on the power supply and are accessible from the back of the Lisa. They are placed here for the user's convenience; there are no internal electrical connections between these controls and the power supply proper. All power output lines are fully isolated from the input AC power by means of transformers or opto-isolators.

### 10.1 Power Supply Block Diagram

The Lisa power supply consists of the following major components shown in Figure 10-2:

- \* AC input circuitry
- \* Flyback oscillator
- \* DC output circuitry
- \* Standby power supply
- \* Video focus/brightness
- \* Front panel control
- \* Safety interlock
- \* Over-temperature monitor.

As can be seen from Figure 10-2, both conducted and radiated noise generated by the switcher are reduced by the input filter on the AC line. The input AC power is then rectified and filtered before being presented to the main flyback oscillator.

The flyback oscillator operates by storing a controlled amount of energy, in the form of a magnetic field, in the switcher-transformer core during the forward cycle as the core is being charged. During the flyback cycle, the core discharges this energy to the load placed across the secondary windings.

The secondary windings are connected to the secondary rectifier and filter circuits which provide the DC power outputs. These outputs are monitored by a regulator circuit, which senses the voltage level and adjusts the power being provided. In addition, there is a crowbar circuit that shuts the power supply down if the output voltages exceed the preset limits.

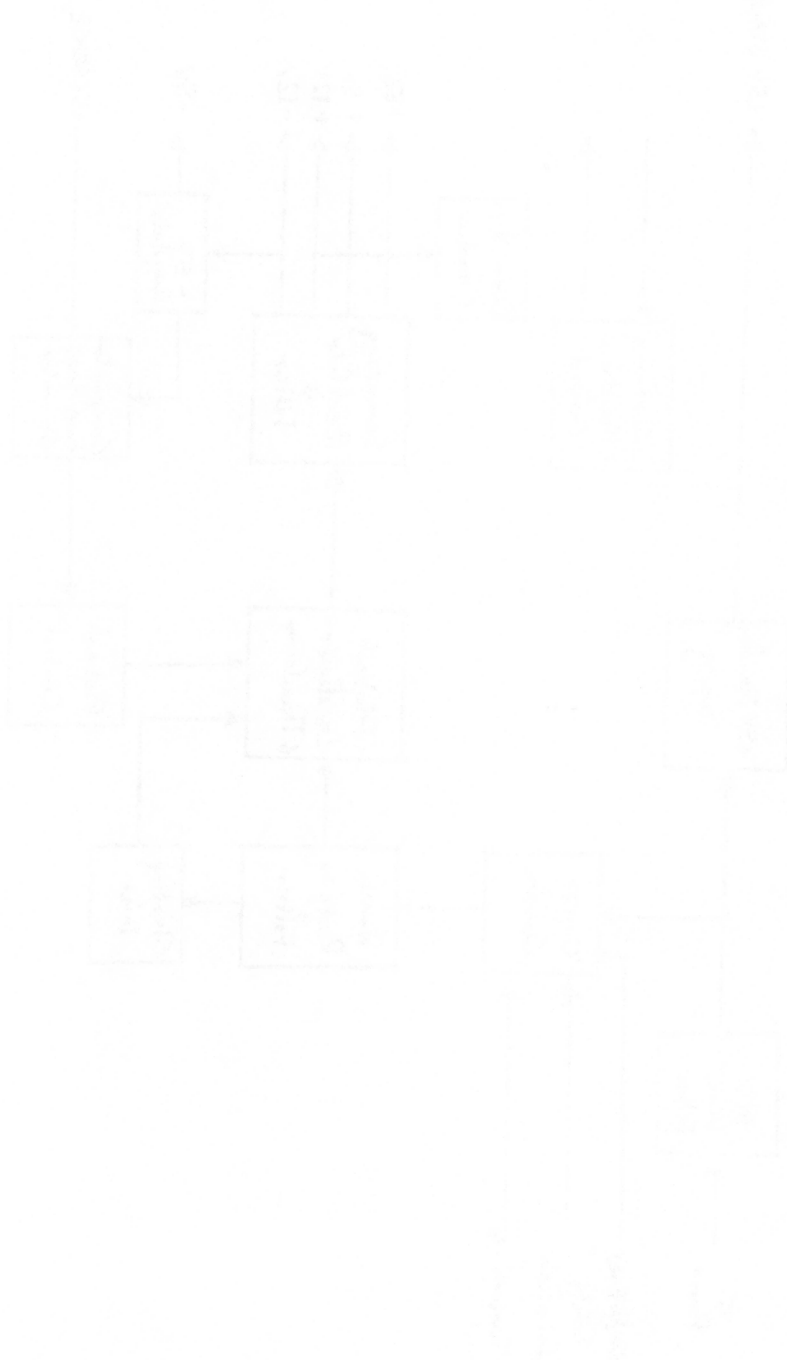


Figure 10-2. Power Supply Block Diagram

#### 10.2 AC Input Circuits

Refer to schematic 050-4011 in Appendix F. The AC input circuits are shown in the upper left quadrant.





### 10.2.1 AC Line Connection

The AC connector P3 supplies power via the power cord and inductor L7 to the J2/P2 connector. This presents the power to the input line filter, which consists of transformer T1, chokes L1 and L2, and capacitors C1, C2, C3 and C4. Thermistor R9 and fusible resistor R8 are provided to limit the inrush current to the power supply.

### 10.2.2 On-Off Control

Power to the supply is controlled by opto-isolator U3, which controls the switching triac CR2. U3 is provided with +5 V from the standby supply via Q5, which biases a light-emitting diode between pins 1 and 16. Pins 8 and 10 operate as a photosensitive Triac. Power is switched to the main supply via CR2.

Resistor R33 is a temperature-sensitive varistor, which biases Q6. Excessive temperature causes Q6 to draw enough current to ground out pin X and to turn off the supply.

The power supply is switched on by applying a logic 1 level to the base of transistor Q5, causing transistor Q5 to conduct, which grounds pin 1 of U3. The power supply switches on and stays on until this level is removed.

A low level on the base of Q5 keeps the power supply in the OFF condition. This can be achieved by either driving it low via the software on-off or by grounding the safety interlock switch.

Warning: Note that the power supply itself remains powered at all times when the Lisa is connected to an AC power source. Test before touching.

### 10.2.3 Primary Rectification

The jumper below R9 is used to select the input AC voltage that is being provided. If 110 V is selected, the connection is made to the midpoint of C12 and C13. Using these capacitors and the diodes CR5 and CR6, the input voltage is doubled using half-wave rectification. The DC voltage output from this rectification and filtering is used to charge up the core of transformer T3 through the 84-turn primary winding between pins 23 and 24.

In the case of a 220 VAC line input, this voltage is full-wave rectified, using diodes CR3, CR4, CR5, and CR6. The resulting rectified DC voltage is again smoothed by the filtering capacitors C12 and C13.

Residual voltage that might be stored in the filtering capacitors when the supply is shut OFF is bled to ground via the large resistance provided by R14 and R15 in series.

### 10.3 The Flyback Oscillator

The focus of the flyback switching oscillator is transistor Q1 and its associated circuitry in the lower left of the schematic. A simplified diagram of the flyback oscillator circuit is shown in Figure 10-3. The points labelled in Figure 10-3 refer to the locations where the waveforms given in the following figures can be observed.

Figure 10-3. Flyback Oscillator Circuit

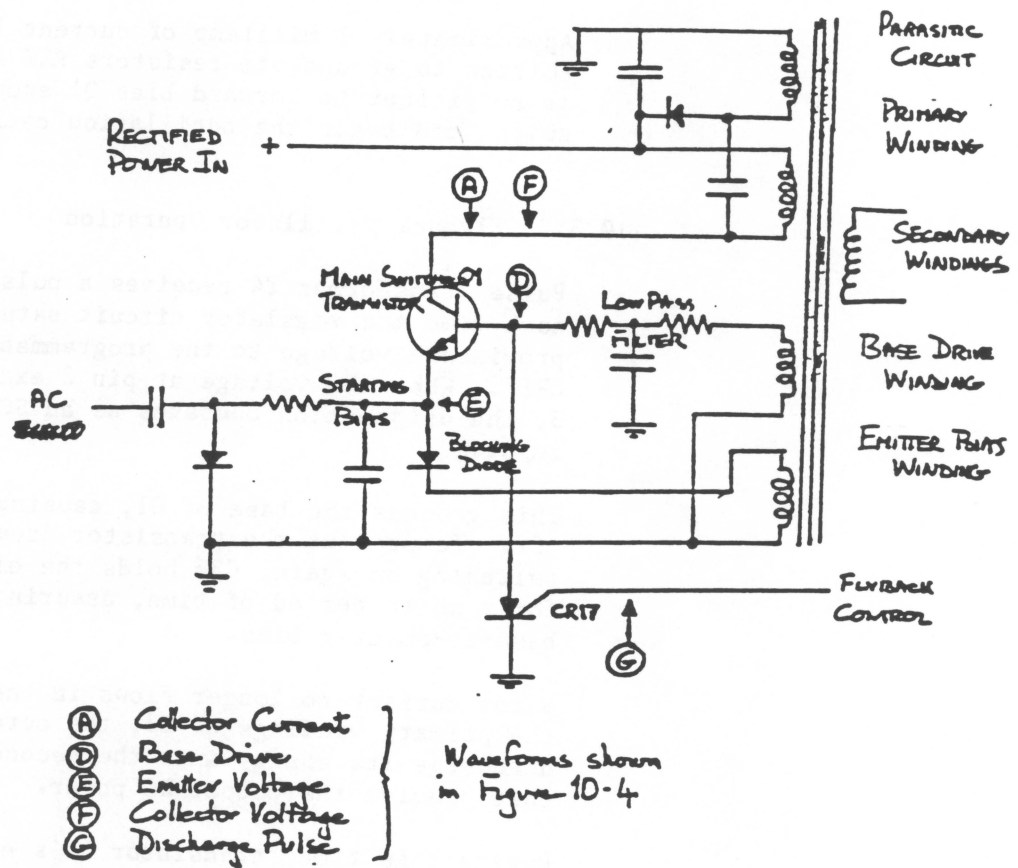


Figure 10-3. Flyback Oscillator Circuit

### 10.3.1 Flyback Starting Bias

In order for the oscillator to begin to oscillate when power is first applied, the emitter of Q1 must be biased negative with respect to its base. This is done by isolation of the base from the emitter by means of diode CR15.

Approximately 1 milliamp of current flows from the emitter to ground via resistors R20 and R21. This is sufficient to forward bias Q1 enough for it to switch and begin the oscillation cycle.

### 10.3.2 Flyback Oscillator Operation

Pulse transformer T4 receives a pulse of current each time the regulator circuit saturates. This provides a voltage to the programmable unijunction CR17. When the voltage at pin 2 exceeds that at pin 3, the unijunction operates as an SCR and shorts the input.

This grounds the base of Q1, causing it to switch off. To prevent the transistor from immediately switching on again, C30 holds the emitter positive for a short period of time, assuring a negative base-to-emitter bias.

Since current no longer flows in the collector and the primary windings of T3, the core begins to discharge its energy into the secondary windings that provide the output DC power.

During this time, transistor Q1's emitter continues to be held positive by the charge in C31, which is discharging through R40/R41 and CR15 to ground via the primary winding 28/22.

Once this charge has leaked away, the current on C25 biases the emitter negative again to provide a starting bias, as explained in Subsection 10.3.1 above, and transistor Q1 begins to conduct. This causes current to flow through the collector and the primary winding 23/24, which induces a magnetic field in the transformer T3 core.

The waveforms at several critical points in the flyback section are shown in Figure 10-4. The letters correspond to the positions where these can be observed.

The parasitic winding of transformer T3 provided by the primary 25/26 is used in conjunction with CR12,

CR23, and C23 to minimize high-voltage transients, which can be generated in T3 and could damage Q1.

Figure 10-4. Flyback Oscillator Waveforms

#### 10.3.3 Flyback Control Circuit

The amount of power output by the supply is a function of the amount of power that is permitted to be stored in the core before the discharge pulse

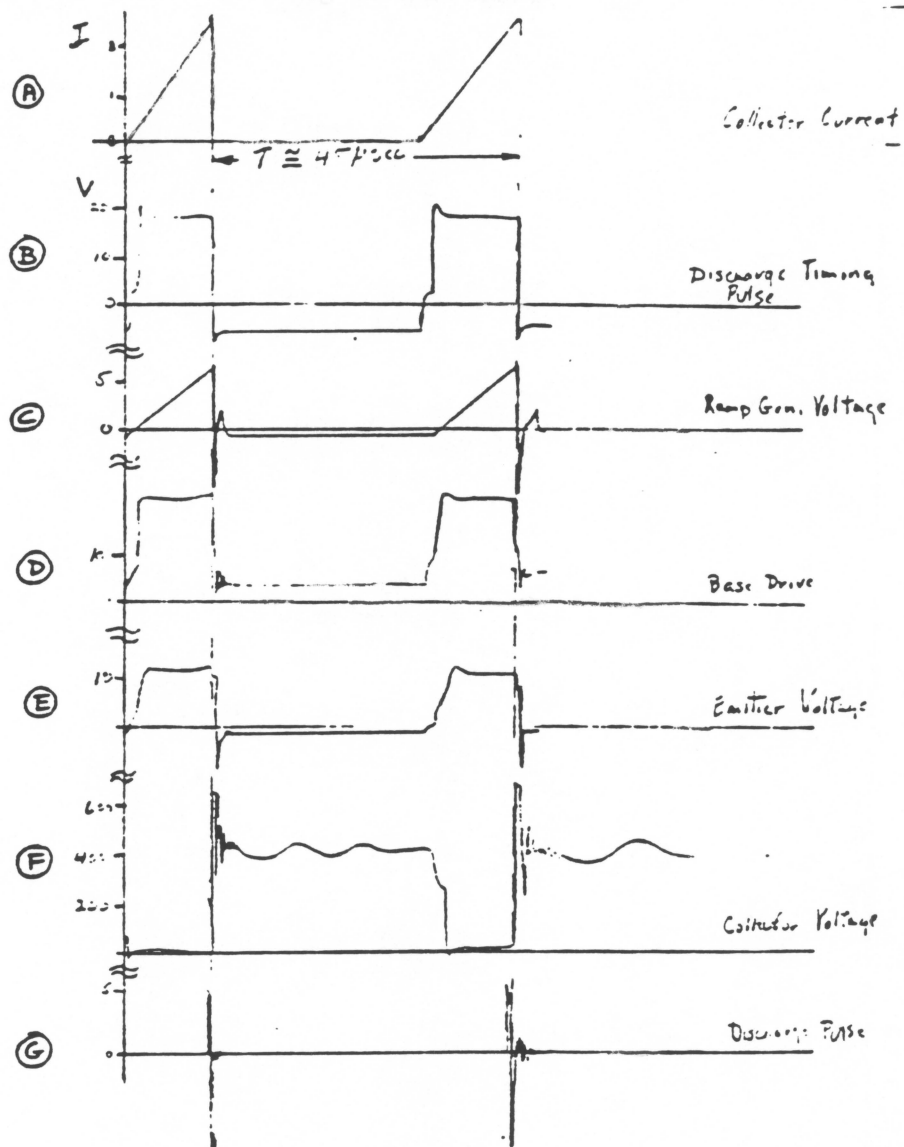


Figure 10-4. Flyback Oscillator Waveforms

transfers it to the secondary windings. The discharge pulse is generated by the circuits at the lower right of the schematic. These are shown in block diagram form in Figure 10-5.



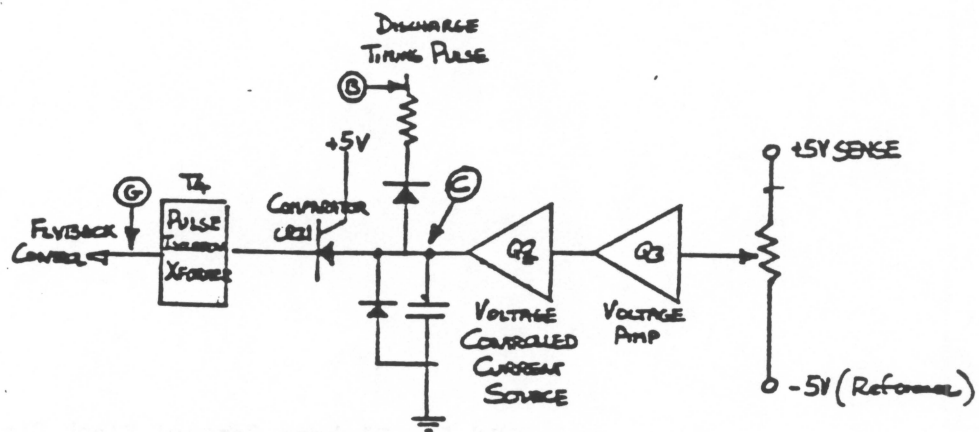
Figure 10-5. Flyback Control Block Diagram

The level of the +5 V line is taken as the measure of all voltage levels being presented to their loads, since all voltages are proportional in the switcher-type supply. This is sensed via pin N of the terminal block and presented to the the top of the voltage adjustment pot R29 via resistors R26 and R27.

A -5 V reference from U2 is fed through resistor R30 and the temperature compensating diode Q4 to the bottom of R29. When the voltages are in balance, the adjustment tap on R29 has a nominal zero volt bias. Otherwise, a voltage bias is applied to the base of Q3, which operates as a voltage amplifier in conjunction with R25 and R32.

The amplified voltage is presented to the base of Q2 through R28. This permits current to flow in proportion to the voltage applied to its base, Q2, operating as a voltage-controlled current source.

The path provided to the -12 V line via CR19 and R24



- (B) Discharge Timing Pulse } Waveforms shown  
 (C) Ramp Voltage } in Figure 10-4  
 (G) Discharge Pulse.

Figure 10-5. Flyback Control Block Diagram



causes the voltage on C34 to start up from zero volts at the same time that transistor Q1 in the main flyback circuit is turned on. The voltage at C34 rises at a rate in proportion to the current being presented to it via Q2.

When the voltage reaches the +5 V level presented to pin 2 of programmable unijunction CR21, the junction conducts. This discharges C34 to ground, which generates a pulse through the secondary winding of T4. This pulse then generates a corresponding pulse in the primary windings, which is used to turn Q1 off, as described above.

The result is a sawtooth waveform at C34. The rate of voltage rise is controlled by the error current from the transconductance amplifier. The total energy stored in the switching transformer T3 is controlled by the rate of voltage rise across C34.

#### 10.4 DC Output Circuitry

The DC output circuitry consists of the secondary circuits to the right of the switching transformer T3 in the schematic. The power supply provides five voltages from the main switcher. These are:

- + 5 V
- 5 V
- +12 V
- 12 V
- +33 V

Of these voltages, the last three originate in their own separate secondary windings. The +5 V supply originates in two windings in parallel, and the -5 V supply is provided by a voltage regulator attached to the -12 V supply.

##### 10.4.1 DC Voltage Outputs

The +33 V supply comes from secondary winding 16/34 on T3, is smoothed by the C9, C10, and L3 network, and is presented on pin D. C11 removes high frequency noise.

The +12 V supply originates in the 20/33 winding, with pin 20 already being biased by the +5 V supply section. It is smoothed by C14, C15, and L4, and presented on pins 5,9,10,E,K, and L. It is also equipped with a crowbar over-voltage protection

circuit, described in subsection 10.4.2. Capacitor C16 removes high-frequency noise on the voltage. The +12 V supply provides the power for the voltage-controlled current source in the oscillator control section. Refer to section 10.3.3.

The +5 V supply is taken from two secondary windings ganged together and passed through the smoothing circuit formed by C18, C19, and C21, with C22 removing high-frequency noise. It is available on pins 11, 12, and M.

The -12 V supply comes from the 17/34 secondary winding of T3 and after smoothing through C27, C28, and L6, it is placed on pin W. The -12 V supply is also used to provide the current being fed through Q2 in the oscillator control circuit; see section 10.3.3.

The -5 V supply is tapped from the -12 V supply by means of the voltage regulator U2. It is smoothed with C24 and presented on pin 19.

Diodes CR7, CR8, CR11, and CR18 prevent reverse power leakage through the transformer during the charging portion of the cycle.

#### 10.4.2 DC Voltage Controls

There are two mechanisms by which the output voltages are monitored in the power supply:

1. Voltages are maintained at the level selected by potentiometer R29 by means of the oscillator control section described in section 10.3.3. Adjustment of the pot raises and lowers all voltages proportionately.
2. Protection for the power supply load is provided by a crowbar circuit attached to the +12 V supply line. The +12 V level is presented to the 12 V zener CR9 through R12. Should the voltage rise enough above +12 V to trigger the diode breakdown, the voltage is applied to pin 3 of the SCR at CR10, causing it to conduct. This shorts the +12 V line to ground, which removes the current source for the oscillator control circuit at Q2. This halts the oscillation, as no firing of CR21 can now take place. Q1 is never switched off and the core ceases to

be repetitively charged, causing all output voltages to decay to zero.

#### 10.5 Standby and Auxiliary Video Circuits

In addition to the main switcher supply itself, two other circuits are present in the power supply. These are described separately because they function independently of the main supply. The other two circuits are the:

- \* +5 V standby supply and
- \* Video focus and brightness adjustments.

##### 10.5.1 The Standby Supply

The standby +5 V supply is a "linear" type and is shown across the top of the schematic. Input AC power from the line filter is presented to the transformer T2. The secondary windings 10/11 and 8/9 each provide approximately 10 VAC 180 degrees out of phase. These are rectified by diodes CR1 and CR22.

The resulting DC voltage has its ripples smoothed by C7 and AC transients removed by C17, before being presented to the 5 V regulator U1. The resulting +5 V output is made available on pin 20.

Note that this line has a low current capacity and is not intended for general use.

##### 10.5.2 The Video Controls

The video controls consist of the resistor network shown towards the upper right of the schematic. Focus is controlled by the potentiometer R3. Brightness is controlled by R5.

Both potentiometers are adjusted by means of the knobs projecting from the back of the power supply. They receive their power from the video board via the +300 V level on pin 1 and the -100 V level on pin B.

be representative of the output of the system.

#### 4.2. The video control system

The video control system is a closed loop system which controls the position of the beam relative to the target in the video field. The system is composed of a video camera, a video amplifier, a video control unit, and a video display unit.

#### 4.3. The video amplifier

The video amplifier is a device which amplifies the video signal from the video camera.

#### 4.4. The video control unit

The video control unit is a device which controls the position of the beam relative to the target in the video field. It receives the video signal from the video amplifier and the target position signal from the target position sensor. It then generates a control signal which is sent to the video display unit.

The video control unit is composed of a video control unit and a video display unit. The video control unit is a device which controls the position of the beam relative to the target in the video field. The video display unit is a device which displays the video signal from the video camera.

The video control unit is a device which controls the position of the beam relative to the target in the video field. It receives the video signal from the video amplifier and the target position signal from the target position sensor.

#### 4.5. The video control system

The video control system is a closed loop system which controls the position of the beam relative to the target in the video field. It receives the video signal from the video amplifier and the target position signal from the target position sensor. It then generates a control signal which is sent to the video display unit.

The video control system is a closed loop system which controls the position of the beam relative to the target in the video field. It receives the video signal from the video amplifier and the target position signal from the target position sensor. It then generates a control signal which is sent to the video display unit.

## CHAPTER 11. ASSEMBLIES

---

The Lisa is engineered to provide maximum serviceability to the user. Refer to section 3.1 for a discussion of packaging. A general view of the assemblies is given in Figures 3-1 and 3-2.

Note that any nuts or screws lost in the cabinet can be retrieved by tipping it forward. They should then slide out of the air vents at the lower front of the cabinet. Removing the bottom panel should not be necessary.

Section 11.1 outlines components that an user can replace. However, the rest of this chapter is concerned principally with components that are not user-serviceable.

### 11.1 User-Replaceable Components

Components that any user can replace on any functioning Lisa include the

- \* Floppy-disk drive assembly
- \* Power supply
- \* Motherboard
- \* Processor board
- \* I/O board
- \* Memory boards
- \* Expansion boards
- \* Keyboard
- \* Mouse.

Access to and replacement of these components is described in step-by-step detail in other documentation. Refer to the Preface for a list of related documents.

Servicing of any other components requires knowledge about high-voltage power supplies and video electronics. Dangerous voltages are present and all service personnel must be trained in their use.

## 11.2 CRT Assembly

Refer to assembly drawing 620-5115 in Appendix I, which shows an exploded view of the cabinet with the CRT assembly. This assembly includes the:

- \* Cathode-ray tube 707-0007
- \* Deflection yoke 159-0007
- \* Video board PCB assembly 620-x121
- \* Video board PCB fab. 820-4012
- \* Flyback transformer xxx-xxxx.

### 11.2.1 CRT Access

Complete the following steps to remove the tube itself. Note that you may need to refer to the Lisa Owner's Guide to complete the first step. Handle the tube with extreme care at all times.

1. Remove the front and back panels from the cabinet, and then remove the power supply and card cage, as shown in the Lisa Owner's Guide.
2. Use a Phillips screwdriver to remove the two screws underneath the back-top of the Lisa.
3. Lift the top of the Lisa up from the back.
4. Unhook the front where it is hooked in place.
5. Remove the top.
6. Remove the white ground lead from the CRT socket. This is attached to the cabinet immediately above the gun.
7. Unplug the CRT socket from the end of the CRT. Take care not to exert too much pressure on the tube itself.
8. Disconnect the two yoke cables, yellow/green and red/blue, from the video board by unplugging the two 6-pin connectors.
9. Pry the plastic cover up from the HV lead that connects to the side of the CRT.

10. Press in one side of the connector under the cover to disengage and remove the lead.
11. The tube itself is held in place by four screws, one at each corner. Prepare a cushioned area on which to place the tube upon removal. Remove the four screws with a 1/4 inch nutdriver.
12. The tube can then be lifted out of the cabinet with the yoke assembly still connected.

To install the tube in the cabinet, complete the steps above in the reverse order. Connect the CRT socket with the CRT almost in place. Connect the HV lead and connectors to the video board after the tube has been screwed in place. Ensure that the correct yoke plugs are inserted in the corresponding sockets on the video board.

#### 11.2.2 Deflection Yoke

Access to the deflection yoke is performed in a manner identical to the CRT. Once the tube has been removed from the cabinet by following the steps in subsection 11.2.1, the deflection yoke can be slid off the back of the CRT by loosening the clamping screw.

Reassembly is performed in the reverse order. Once the deflection yoke has been changed, readjustment of picture orientation by means of the video board controls might be necessary.

#### 11.2.3 Video Board

Access to the video board is performed first by removing the CRT, as outlined in subsection 11.2.1. The procedure is then continued as follows:

1. Disconnect the cabling to the flyback transformer from connector P3 on the video board.
2. Release the CRT socket from the two cable tiedowns.
3. Use a Phillips screwdriver to remove the screws that hold the top of the video



board onto the back wall of the CRT enclosure.

4. The board can now be removed from the connector on its lower edge and removed from the system.

To install a video board, perform the same steps in reverse order. A new video board does not have the correct adjustments for contrast and focus. Once the Lisa is reassembled and operational, adjust controls at the back of the power supply until the picture is acceptable.

If you replace a video board, picture-size and centering adjustment might be required. This is accomplished with controls on the board itself.

#### 11.2.4 Flyback Transformer

The flyback transformer is attached to the floor of the tube enclosure. It is removed by performing the following steps:

1. Remove the CRT by following the instructions in subsection 11.2.1.
2. Disconnect the flyback transformer cable from P3 on the video board.
3. Remove the flyback transformer cable from any cable tiedowns.
4. Remove the bottom panel of the cabinet by removing the six screws in the logic enclosure.
5. Remove the two nuts underneath the cabinet, taking care to hold the transformer to prevent it falling once these are removed.

When replacing the assembly, ensure that the HV lead is to the back. Clip the bottom of the transformer at the front first and follow the disassembly procedure in the reverse order.



### 11.3 Cabling

Due to the modular nature of the logic electronics, the interconnections within the Lisa have been kept to a minimum. Interconnections fall into four categories:

- \* Power cabling
- \* Disk cabling
- \* Interface connections
- \* Logic connections.

#### 11.3.1 Power Cabling

Power is supplied to the Lisa through a three-wire wall connector and a power cable to a socket in the power-supply assembly. The power-supply assembly in turn provides power to the Lisa via an edge connector, which mates into a plug in the CRT enclosure. The edge connector is part of the power cabling harness that distributes power to the following assemblies:

- \* Motherboard
- \* Video board
- \* Speaker
- \* Cover interlock.

Should it become necessary to remove the power harness, the procedure in subsection 11.2.1 must be followed to first remove the CRT. The following steps are then followed:

1. Remove the two screws that hold the video board connector to the wall of the CRT enclosure.
2. Remove the two screws that hold the power supply connector plug to the back wall of the CRT enclosure.
3. Disconnect the speaker plug on the floor of the CRT enclosure.
4. Remove the screw that holds the cover interlock to the cabinet beside the speaker.
5. Remove the two screws that hold the motherboard connector to the wall of the logic board enclosure.

6. Remove any tiedowns holding the cabling to the cabinet.
7. The harness can then be removed from the cabinet.

Installation of a power cable harness is performed in the reverse order. Before installing any tiedowns, ensure that the cable is correctly positioned and that all connectors can reach their proper locations.

If difficulty is encountered mounting the motherboard connector, removal of the flyback transformer makes this connector more accessible.

Ensure that the ground connectors are correctly attached by one of the mounting screws for both the power supply and the video board connectors.

#### 11.3.2 Disk Cabling

The two floppy-disk drives are installed as a single unit in the right-front of the cabinet. They are connected to the logic by means of flat ribbon cables, one for each drive. The cable also provides the logic connection to the board that carries the power switch and the keyboard phone jack at the lower-right front of the cabinet.

To gain access to the cable, first remove the drive assembly 620-5107. Then all steps called for in subsection 11.2.1 must be performed to remove the card cage, power supply and CRT. Finally, to remove the disk cable:

1. Release the cable from the two tiedowns.
2. Unplug the connector to the switch board that contains the power switch and the keyboard jack.
3. Remove the two screws that hold the disk cable motherboard connector to the back of the logic enclosure wall.

Installation is performed in the reverse order. Ensure that the cable has sufficient slack to allow the drives to be placed in front of the cabinet with the cable attached.

### 11.3.3 Interface Connections

Interface cabling in the Lisa consists of those connectors at the back of the motherboard. No disassembly procedure is required. It is however recommended that connections are made and broken with power off.

The interface connectors themselves are soldered to the motherboard and are not separately replaceable.

### 11.3.4 Logic Connections

All logic cabling in the Lisa is made by means of edge connectors between logic cards, with the exception of the disk cable, described in subsection 11.3.2 above. There is therefore no disassembly procedure.

## 11.4 Disk Drive Assembly

The disk drive assembly consists of the following components:

- \* Drive carrier 805-4014
- \* Drive-carrier shelf 805-4013
- \* Upper-disk drive 653-6110
- \* Lower-disk drive 653-6110.

There is no difference between the upper and lower drives. They are distinguished logically by the connections made by the two plugs on the disk drive cable.

To remove the bottom drive, remove the four 6-32 Phillips screws on the under surface of the assembly and slide the drive out.

To remove the top drive, remove the six 6-32 Phillips screws that hold the drive carrier shelf in the middle of the assembly, there are three on each side. The shelf and top drive can then be slid out of the assembly. The drive is detached by removal of the four screws on the underside.

## 11.5 Miscellaneous

The following assemblies are present in the cabinet in addition to those covered in the Lisa Owner's Guide and

the preceding sections. These are:

- \* Speaker assembly
- \* On-off switch assembly
- \* Motherboard assembly
- \* Cabinet assemblies.

#### 11.5.1 Speaker Assembly

The speaker is located on the floor of the CRT enclosure. In order to remove it, it is necessary to remove the CRT, as described in subsection 11.2.1. Then the following procedure is followed:

1. Disconnect the speaker from the power cable harness by unplugging the plastic connector.
2. Remove the three screws holding the speaker to the cabinet.
3. Remove the speaker.

To install a speaker, follow the steps above in the reverse order.

#### 11.5.2 On-Off Switch Board Assembly

The on-off switch board is located on the lower-right front of the cabinet. In order to remove the on-off switch board, the disk drive assembly must first be removed, as explained in section 11.4. Then do the following:

1. Unplug the disk cable from the connector on the switch board.
2. Use a long-reach Phillips screwdriver to remove the two screws which hold the board to the base of the cabinet.
3. Remove the board.

Reassembly is performed in the reverse order.

#### 11.5.3 Motherboard Assembly

The motherboard assembly is the carrier unit for all the logic cards in the system. It consists of the

basic PCB with card edge connectors and interface connectors attached plus card guides and a metal frame.

The top of the expansion board card guides can be removed by removing the three Phillips screws that hold it to the top of the left-hand main card guide.

The card guides can be detached by removing the six 4-40 Phillips screws that hold the main card guide assembly in place. The two main card guides are attached to one another by three support bars, each of which is held at each end by a Phillips screw.

The metal motherboard frame can be detached from the motherboard itself by removing the fifteen 4-40 Phillips screws that hold the two together. Eight of the screws hold the interface connectors in place. The other seven hold the motherboard to the frame.

## 11.6 Keyboard and Mouse

The keyboard and mouse can be detached from the cabinet, since they are physically separate elements. Also, both may be disassembled if the need arises.

### 11.6.1 Keyboard Assembly

The keyboard consists of a single PCB which carries all the keys sandwiched between two halves of the case. To access the assembly:

1. Disconnect the keyboard from the Lisa by removing the keyboard jack from the plug at the lower-right of the cabinet.
2. Using a Phillips screwdriver, remove the five screws that hold the keyboard together.
3. Remove the keyboard PCB and place it on a flat surface.

Reassembly is performed in the reverse order. Care must be taken not to apply excessive torque to the threads of the screws.

The keyboard cable is held in place by a Molex-type connector, which provides strain relief. This can

be removed as required.

#### 11.6.2 Mouse Assembly

The mouse is a stand-alone device that has a few moving parts. To access the mouse assembly, unplug it from the back of the cabinet and remove the three plastic screws on the underside that hold the body together.

GLOSSARY

---

**Acknowledge**

A signal used during handshake operations to indicate that the current step has been completed.

**AS**

Abbreviation for address strobe; <<which is ....>>.

**Asserted**

A signal in a true state, which means that the term SIGNAL is in a "high" or "1" state; the term SIGNAL/ is in a "low" or "0" state if it is asserted.

**Asynchronous modem**

A modem that handles asynchronous transmissions. In asynchronous communication, each character is transmitted with its own framing information telling the receiver where the character starts and stops. Since each character is a complete message, the time interval between successive characters need not be fixed.

**Autovector**

The jump to the interrupt handler preloaded at a certain location. When handling certain interrupts, the 68000 automatically jumps to a location predefined for the given interrupt.

**Battery backup**

The rechargeable Nickel/Cadmium batteries with which the I/O board is equipped. Whenever the Lisa is disconnected from a power source, the COPS device continues to receive power from these batteries to allow it to provide a true real-time clock and also initiate software-controlled power-up.

**Baud rate**

The rate at which a modem sends and/or receives information. 110



baud means the modem is handling approximately 110 bits per second. If there are two stop bits, a start bit, a parity bit, and a seven bit ASCII character code, 110 bits per second translates into about 10 characters per second.

**BCD**

Abbreviation for binary coded decimal; <<which is ....>>.

**Bit**

Acronym for binary digit; an item of data with only two possible states, either 1 (one) or 0 (zero).

**B/L/**

Abbreviation for base limit; <<which is ....>>.

**Block**

A contiguous set of data in the CPU; normally a unit of data written to and read from disk (524 bytes). When referring to memory, it may be of any length, but all addresses of the block are contiguous.

**Bootling**

The process of getting the operating-system software into place and executing. When a computer is turned on, it must "bootstrap" itself into a useable state.

**Buffer**

A logic device used as a bus driver, whether it has latch capability or not.

**Bus**

A set of parallel wires, traces, and paths that carry related data and control information from one device to another.

**Bus Timeout**

A feature that permits the CPU to give peripherals a variable amount of time in which to respond to a transfer request. If the peripheral does not respond to a cycle after approximately 30 microseconds, a bus timeout occurs and the CPU logs an error.

**Byte**

A group of bits. On the Lisa, a byte is always 8 bits.

**Byte parity**

<<Is ....will someone define it concisely? Following is what the



author provided>> When bytes of data are being moved around, one or more bits in the byte can get improperly flipped. These incorrect bits can sometimes be detected by checking the byte parity. The byte's parity is odd if there is an odd number of 1 bits in the byte. If another bit is available in the byte, the sender can ensure that every byte has even parity upon being transmitted. The receiver can then check each byte's parity, and if any are odd, it can inform the CPU that something has gone wrong.

#### Card cage

The metal frame that holds the printed circuit boards in the cabinet.

#### CAS

Abbreviation for column address strobe; <<which is ....>>

#### CASEN

Abbreviation for case enable; <<which is ....>>

#### Checksum

Similar to byte parity; a number used to ensure that data has not suffered degradation during transfer. A checksum is usually generated by addition of all the bytes in a block in a certain pattern. It is written at the end of blocks of data written to disk.

#### Clock

A continuous, regular waveform used to control the timing of logic decisions.

#### CMOS

Abbreviation for complementary metal-oxide semiconductor. CMOS combines N-channel and P-channel MOS transistors to give low power consumption. Since the non-volatile parameter memory must not consume much power, it is implemented with CMOS chips.

#### Context

A complete set of 128 pairs of registers within the MMU; the Lisa is configured to operate in one of four contexts. The operating system normally executes in context 0. Switching between programs can be

quickly done by changing context.

**COPS**

Abbreviation for control-oriented processor system; in the Lisa, it is used as a controller at both ends of the keyboard interface.

**Counter**

A logic device that sequentially increments or decrements each time a clocking pulse occurs.

**CPU**

Abbreviation for central processing unit; the Motorola 68000 16-bit general-purpose device in the cabinet.

**CRT**

Abbreviation for cathode-ray tube; the screen inside the cabinet.

**Cycle**

The interval between the same phase position of two adjacent clock pulses.

**Daisy chain**

A method of connecting several devices to a single I/O port. Signal lines are passed from device to device in a chain.

**Deasserted**

A signal in a false state, which means that the term SIGNAL would be deasserted if it is "low" or "0"; the term SIGNAL/ is deasserted if it is "high" or "1".

**Decode**

Synonym of select. The decoder's input address determines which of its many outputs is asserted.

**Disable**

Holding any signal in a deasserted state, irrespective of other inputs.

**DMA**

Abbreviation for direct memory access; a device can read and write memory locations directly, without any intervention from the CPU. However, normal memory access goes through the 68000 and its memory manager.

**Driver**

A logic device capable of driving signals that have a large electrical load, such as occurs on a bus.

**D/A**

Abbreviation for digital to analog.

## ECC

When a digital signal is used to control an analog device, the bits in the digital word must be converted into analog voltage levels.

Abbreviation for error correction code; in memory, ECC is a way to regenerate erroneous data bits that occur. Not yet implemented.

## Edge detector

Logic that is capable of detecting a signal transition high-to-low or low-to-high.

## Enable

Allow a signal to either respond to its gating inputs or to gate data into a further stage of the logic.

## Expansion bus

The signal subset of the internal system bus that is made available on three card slots in the cabinet for the addition of other logic components.

## Flag

A status bit that indicates the occurrence of some condition in the Lisa. It is usually available for interrogation by the CPU.

## Flip flop

A digital circuit used to store one bit of data. There are variants on the basic theme but generally, the state changes synchronously with a clock edge, depending on the input state. Also known as "flop" and "FF".

## Format

An exact description of the sequence of bits as they are organized on a disk.

## Gate

A switch that controls the flow of data according to some Boolean function of its inputs.

## GOBYTE

The flag byte used to tell the floppy-disk controller that the CPU wishes to have a macro instruction executed.

## Handshake

The control of data-transfer between devices. Each device has a way to tell the other that its side of the operation is complete. For example,

a processor writes data to a register, and then signals a device that data is ready to be read. The device reads the data, and then sends the processor a signal that it has finished reading and is ready for more. The handshake insures that the processor does not write new data to the register, thereby destroying the old data, before the device has had a chance to read the old data.

### Hang

What happens to the computer when the CPU goes into an infinite loop or a wait state. If the operating system is unable to recover, your only recourse is to restart the machine.

### Hard error

A non-recoverable error.

### Header

The series of bytes at the beginning of a sector on a disk that identifies the sector.

### Hexadecimal

The number system used within the Lisa. The digits in this base-16 system are 0-9 and A-F. All addresses and data are expressed in terms of hexadecimal numbers

### High

A voltage state that can signify either true or false, depending on the logic being used.

### Horizontal retrace

The period of time when the electron beam in the monitor is returning from the end of a line to begin the next.

### KByte

Abbreviation for kilobyte; equal to 1,000 bytes.

### KHz

Abbreviation for kilohertz; equal to 1,000 cycles per second.

### Kword

Abbreviation for kiloword; equal to 1,000 words.

### I/O

Abbreviation for input and/or output; a generic term that describes the peripheral system with which the CPU communicates with the outside world or with data storage other than main memory.

## IOB

Abbreviation for I/O block; a block of memory used to control and communicate with the floppy-disk controller.

## Interrupt

A signal that causes the CPU to suspend its current operation and ....

## Latch

A register that contains data for a transient period of time; usually until read.

## Logical Address

An expression that describes the address space generated by the CPU. Since logical addresses are translated into physical addresses by the MMU, these are not physical addresses. They can therefore remain constant even if physical addresses change, provided the MMU is informed of the change.

## Low

A voltage state that can signify either true or false, depending on the logic being used.

## LSB

Abbreviation for least significant bit; <<which is ....>>.

## ma

Abbreviation for milliamperes; equal to one-thousandth of an ampere.

## Main memory

The RAM located on the memory boards in the cabinet. Main memory is used as storage by the CPU only, and can vary in size from 256 Kbytes to 2 Mbytes.

## MALE

Abbreviation for memory address latch enable; <<which is ....>>.

## Memory bus

The collection of signals that interface the memory boards with the processor board.

## Map

The translation of one addressing system into another. In the Lisa, it describes the process of converting logical addresses into physical addresses.

Mask	A pattern of bits that controls the contents of a register. Mask bits need not be contiguous.
Matrix	The two-dimensional array of bits in memory.
Mbyte	Abbreviation for megabyte; equal to one million bytes.
MHz	Abbreviation for megahertz; equal to one million cycles per second.
mm	Abbreviation for millimeter; a unit of length equal to one-thousandth of a meter.
MMU	Abbreviation for memory management unit: hardware in the Lisa that provides a feature called relocation. This allows the CPU and, therefore, resident software to operate on logical addresses, which are translated into physical addresses by the MMU.
Motherboard	The circuit board whose main function is to provide connection between the other circuit boards.
Mouse	A detachable device that you can roll across a flat surface to direct cursor movement on the screen.
ms	Abbreviation for millisecond; equal to one-thousandth of a second.
MSB	Abbreviation for most significant bit; <<which is ...>>.
mV	Abbreviation for millivolt; equal to one-thousandth of a volt.
N-key rollover	Nearly all keyboard interfaces work by scanning the keys and forming a two-dimensional matrix representing the state of the keys. The logic involved can tell when two keys are being pressed simultaneously, rollover; however, when three or more keys are held down, phantom keys can appear. The N-key rollover design adds a diode in series with every key switch to eliminate hidden paths.

## Nibble

A set of bits smaller than a byte. On the Lisa, a nibble is 4 bits.

## NMI

Abbreviation for non-maskable interrupt. When the processor receives an interrupt, it usually checks a mask to see whether it should pass control to that interrupt's handler. A non-maskable interrupt is always honored.

## ns

Abbreviation for nanosecond; equal to one-billionth of a second.

## Oscillator

A device that provides an accurate time period. It is usually used in clock generation.

## Page

Defines a contiguous area of main memory that is 512 bytes long. A logical page is this size of area within the CPU's logical address space. A physical page is this size of area within the main memory.

## Parameter memory

A non-volatile block of RAM set aside for such things as the system serial number, configuration data, and user-defined information.

## Parity

A function of the number of bits in a word that are high. If there are an even number, the word has even parity. Parity can be checked to insure that one bit in a word has not been incorrectly flipped during transmission.

## PC

Abbreviation for program counter. A register within the CPU that keeps track of the next instruction to be executed after the current one is complete.

## PIA

Abbreviation for peripheral interface adapter; an LSI logic device that assists the CPU in interfacing to peripheral logic. On the Lisa, Motorola 6522 VIA devices are used.

## Pixel

The smallest controllable unit of area on the CRT. It corresponds to one bit of video data and can only be



on or off, black or white.

Port

An I/O address location within I/O space. In general, any group of addresses within the same PIA is termed a port also.

PROM

Abbreviation for programmable read-only memory; it refers to the fact that the ROM can be programmed. It is usually not-alterable while it is in a computer.

RAM

Abbreviation for random-access memory; it is read/write memory.

Recalibrate

A technique of establishing absolute track position of the disk heads returning to a known position.

Refresh

The operation whereby the dynamic RAM's used for data storage are refreshed with the data they contain.

Relocation

The technique implemented in MMU to map the logical address space used by the CPU into the physical address space that reflects the actual configuration of the Lisa.

Retry

Repeat the execution of an instruction or routine in an attempt to avoid an error result. Used most frequently on disk access.

ROM

Abbreviation for read-only memory; devices that are fabricated with an unalterable program already present.

RS232-C

An interface standard that defines a bit-serial interface for use in conjunction with a modem, plus a corresponding communication protocol.

RTC

Abbreviation for real-time clock; it is a function implemented on I/O board by the COPS device. Since power is always available to this device, it is used to calculate the real elapsed time at all times. This is available to software for interrogation.



**RWTS****Abbreviation for**

Read/Write/Track/Sector; it refers to the controlling routines that drive the floppy disk. The RWTS is that part of the floppy-disk controller's resident program that interrupts the CPU upon completion since it involves actual disk control.

**Sector**

The smallest addressable portion of a disk. Each track normally contains several segments.

**Seek**

To move the heads across the disk surface to position them above a particular track.

**Segment**

An area of the system-address space. It is the main unit of memory dealt with by the MMU. A logical segment is a contiguous block of 128 Kbytes in the CPU's logical address space. It consists of 256 logical pages. A physical segment consists of between 1 and 256 physical pages, depending on the size allocated to it by software via the MMU. Each logical segment has a corresponding physical segment. There are 128 of each in a Lisa.

**Sense**

The operation of interrogating status.

**Shared Memory**

An area of memory that can be accessed by more than one processor. In a Lisa, the disk controller storage area is shared between the 6504 and the CPU.

**Slot**

One of the three slots in the cabinet available for peripheral boards.

**SLR**

Abbreviation for segment limit register; one of the registers within the four sets of 128 such registers contained in the MMU. It is used to define both the size of the physical segment being accessed in one page increments and the type of storage being accessed.

**SOR**

Abbreviation for segment origin

register; one of the registers within the four sets of 128 such registers contained in the MMU. It is used to define the lowest address of the physical segment being accessed in terms of pages from the lowest address in memory.

**State Machine**

A collection of logic devices capable of executing a very simple series of steps sequentially. It is the simplest form of computer.

**Status**

An array of bits that is available to the CPU to inform it of the state of certain portions of the Lisa.

**Strobe**

A signal that indicates that data is being transferred for the time that the strobe signal is asserted.

**Sync**

A synchronization signal that permits other signals to assume a known state at a known time.

**Synchronous modem**

A modem that transmits synchronously, which puts the framing information around a group of characters. The transmitter then automatically inserts fill characters into the stream whenever necessary to maintain synchronicity. Because more of the bits are data, there are fewer stop and start bits than in asynchronous transmission, data transfers can go at a faster rate.

**System Bus**

The main bus, which is used to communicate between the processor and I/O boards. A subset of it extends to the expansion slots and is called the expansion bus.

**Tristate**

A logic output that can be inactive, asserted or deasserted. These three states may also be termed active high, active low, and open.

**TTL**

Abbreviation for Transistor-Transistor Logic; the most common kind of digital device in the Lisa.

**Vertical retrace**

The period of time during which the

electron beam in the monitor is returned from the bottom of the screen to the top.

VIA	Abbreviation for Versatile Interface Adapter; the name used by Motorola for their 6522 peripheral port control devices.
Video	Of or pertaining to the CRT and its control circuitry.
Word	A group of bytes. On the Lisa, a word is usually 16 bits, or two bytes.
Write Protect	An operation that inhibits any writing operation to the item protected. This technique can be used for both memory and disks.
Yoke	The control windings on the CRT that deflect the beam.
ZIF	Abbreviation for zero insertion force, which refers to the type of connectors used for the three expansion slots in the cabinet. Since the slots are accessed from the side, a locking device is used to open the connector and permit the PCB's to be slid in from the side of the connector before being locked in place.

...the ...  
...the ...  
...the ...

...the ...  
...the ...  
...the ...

...the ...  
...the ...

...the ...  
...the ...

...the ...  
...the ...  
...the ...

...the ...  
...the ...

...the ...  
...the ...  
...the ...

...the ...  
...the ...  
...the ...  
...the ...

...

...

...

...

...

...

...

...

Appendix A. Processor Board Schematic

---

Paula Nunez will supply camera ready copy.



Appendix B. Memory Board Schematic

---

Paula Nunez will supply camera ready copy.

1950-1951

1950-1951



Appendix C. I/O Board Schematic

---

Paula Nunez will supply camera ready copy.

Amending C. 110 to read:

Paul: House will amend the bill to read:

Appendix D. Video Board Schematic

---

Paula Nunez will supply camera ready copy.

1. The theory of operations

2. The theory of operations

3. The theory of operations

Appendix E. Keyboard Schematic

---

Paula Nunez will supply camera ready copy.

APPENDIX A. LINEAR ALGEBRA

Let  $V$  be a vector space over  $F$ .

Appendix F. Power Supply Schematic

---

Paula Nunez will supply camera ready copy.





Appendix G. Motherboard Schematic

---

Paula Nunez will supply camera ready copy.

1954

1954

1954

## Appendix H. Alternate Keyboard Layouts

---

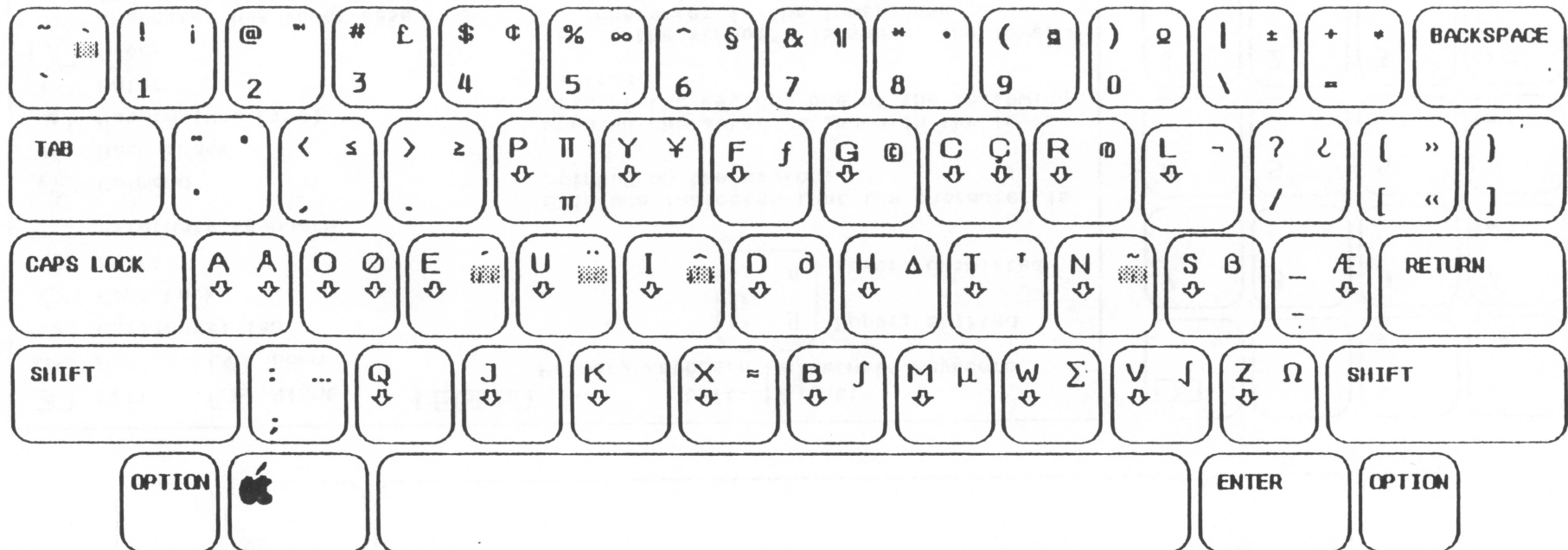
Following Lisa keyboard layouts are available in addition to the standard, North American, layout shown in Chapter 8.

REPORTS OF THE AMERICAN MEDICAL ASSOCIATION

Published by the American Medical Association, 535 North Dearborn Street, Chicago, Ill. 60610  
Subscription price, \$10.00 per annum in advance. Single copies, 25 cents. Entered as second-class matter, May 2, 1917. Postage paid at Chicago, Ill. and at additional mailing offices. Postmaster: Send address changes in this journal to THE JOURNAL OF THE AMERICAN MEDICAL ASSOCIATION, 535 North Dearborn Street, Chicago, Ill. 60610.

# US DVORAK Keyboard Layout

TN / 8 Nov, 82



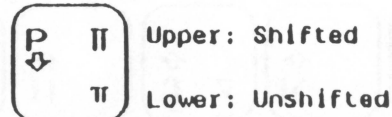
## LEGEND

- Left
- Right
- Up
- Down

- Dead key, no character is generated until next key is pressed.
- Character has lower case and is affected by Caps Lock.

OPTION Selects Alternate Keyboard

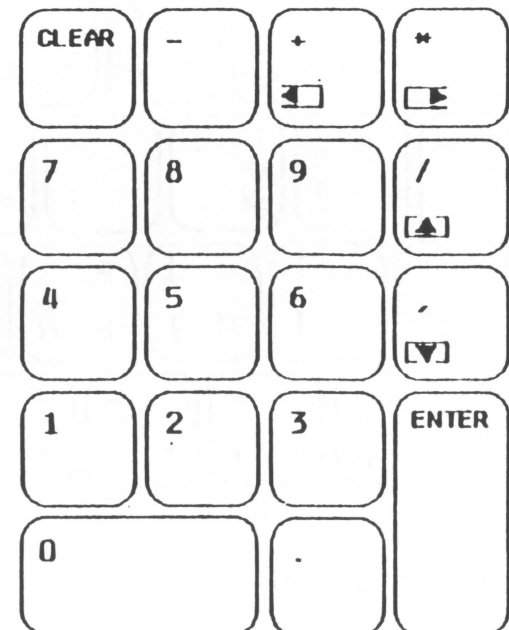
Left: | Right:  
Primary keyboard | Alternate keyboard



**Boldface** indicates that the character is printed on the keytop.

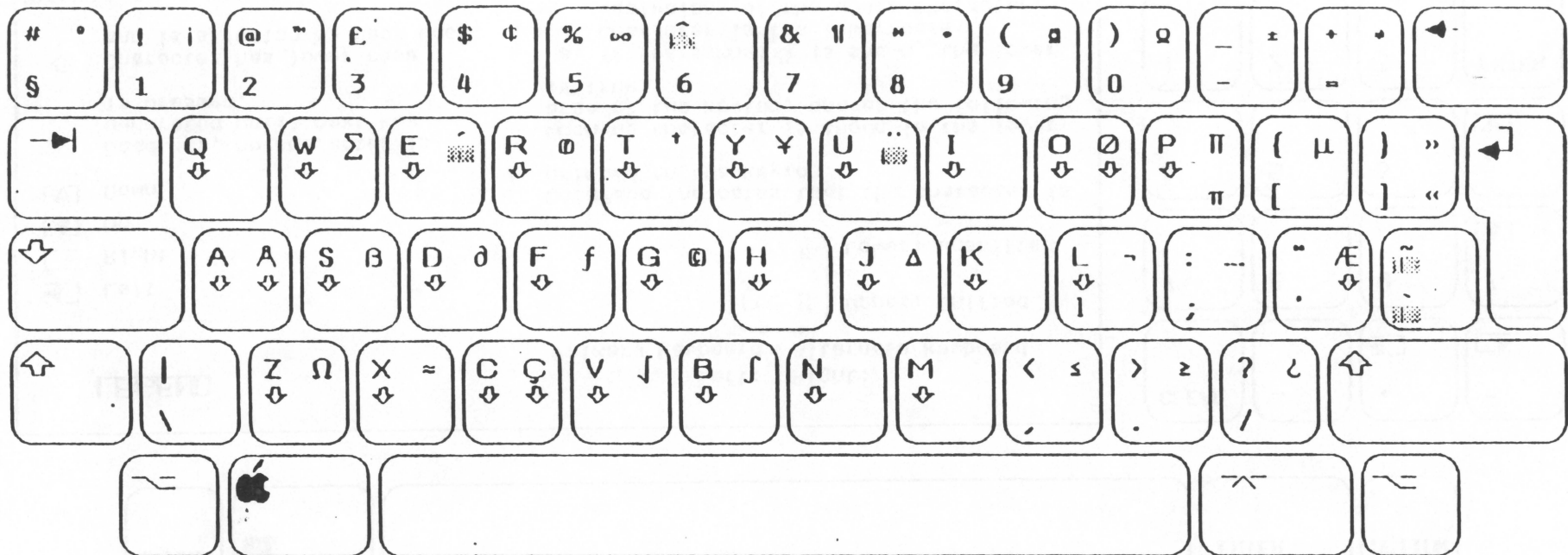
When no character is shown in the lower area of the keytop, one of the following is true:

- a. If the symbol is shown, the lower character is the lower case equivalent of the upper character.
- b. If the symbol is not shown, the lower character is the same as the upper character.



# UK Keyboard Layout

TN / 8 Nov, 82



Left

Right

Up

Down

Horizontal Tab

Caps Lock

Shift

Alternate Keyboard

Command

Back Space

Return (New Line)

Enter

Clear

Character has lower case and is affected by Caps Lock.

Dead key, no character is generated until next key is pressed.

### LEGEND

Left: Primary keyboard

Right: Alternate keyboard

Upper: Shifted

Lower: Unshifted

**Boldface** indicates that the character is printed on the keytop.

When no character is shown in the lower area of the keytop, one of the following is true:

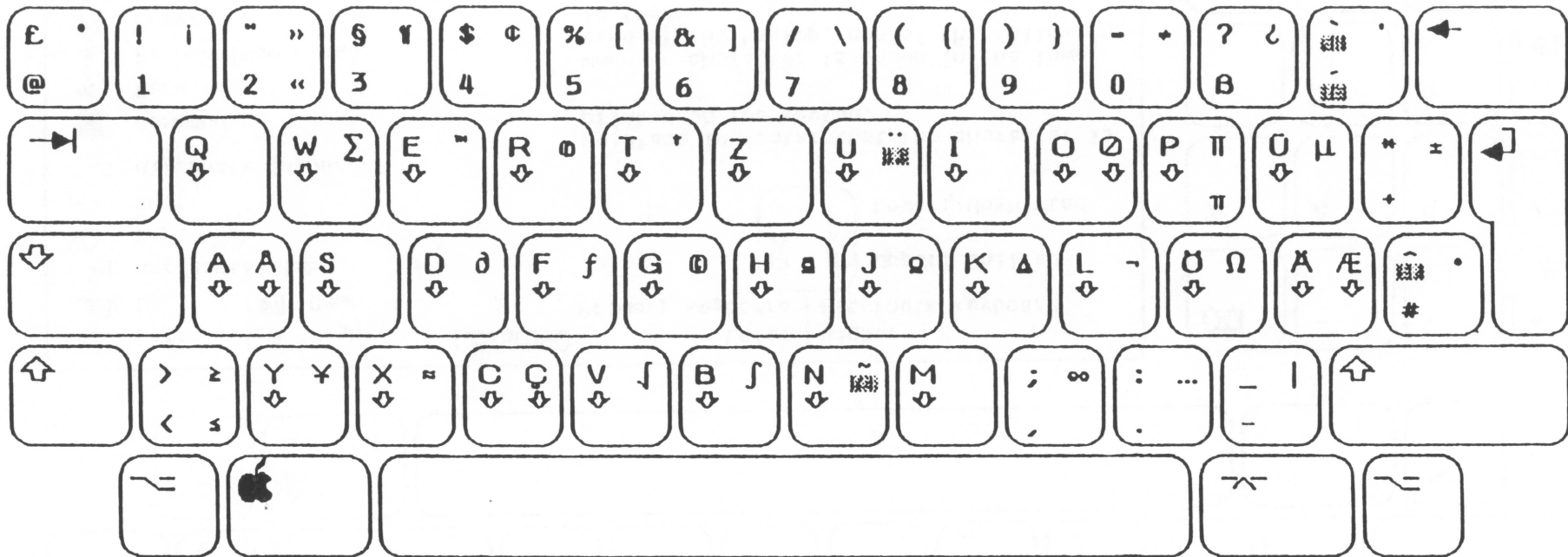
- If the symbol is shown, the lower character is the lower case equivalent of the upper character.
- If the symbol is not shown, the lower character is the same as the upper character.

The diagram shows the right side of the keyboard with the following key assignments:

- Row 1:** Clear, -, +, \*
- Row 2:** 7, 8, 9, /
- Row 3:** 4, 5, 6, ,
- Row 4:** 1, 2, 3, ~
- Row 5:** 0, ., Command

# German Keyboard Layout

TN / 8 Nov, 82



⇧ Left	⇧ Right	<b>LEGEND</b>	Left:	Right:
⇧ Up	⇧ Down		Primary keyboard	Alternate keyboard

**P** **Π** Upper: Shifted  
**q** **π** Lower: Unshifted

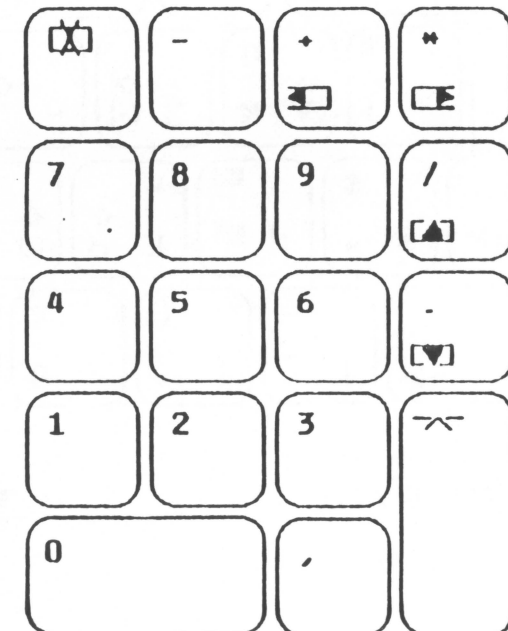
**Boldface** indicates that the character is printed on the keytop.

When no character is shown in the lower area of the keytop, one of the following is true:

- If the symbol **q** is shown, the lower character is the lower case equivalent of the upper character.
- If the symbol **q** is not shown, the lower character is the same as the upper character.

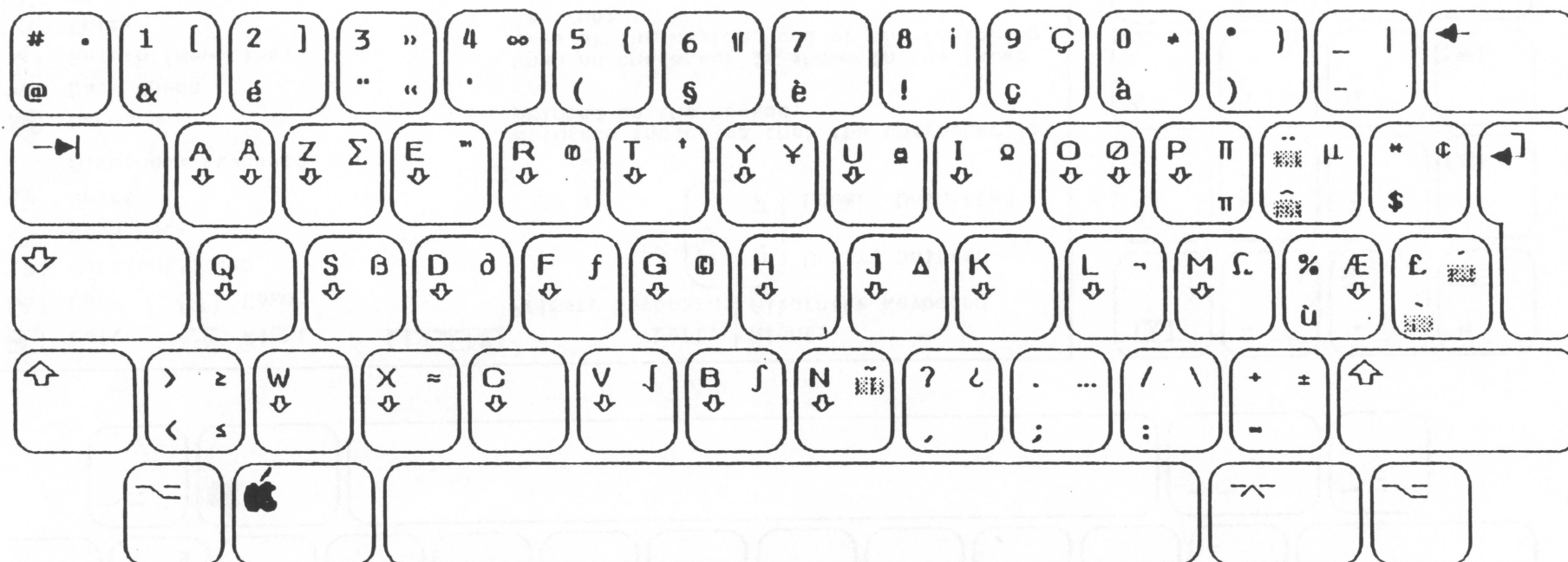
**⇧** Character has lower case and is affected by Caps Lock.

**~** Dead key, no character is generated until next key is pressed.



# French Keyboard Layout

TN / 8 Nov, 82



	Left		Right
	Up		Down
	Horizontal Tab		
	Caps Lock		
	Shift		
	Alternate Keyboard		
	Command		
	Back Space		
	Return (New Line)		
	Enter		
	Clear		
	Character has lower case and is affected by Caps Lock.		
	Dead key, no character is generated until next key is pressed.		

**LEGEND**

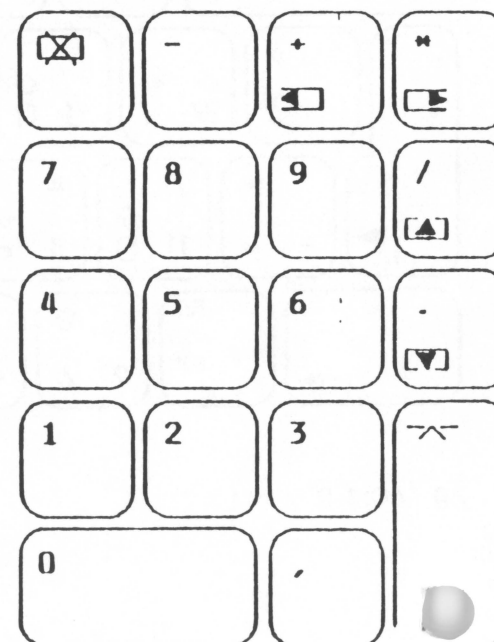
Left:	Right:
Primary keyboard	Alternate keyboard

P	Π	Upper: Shifted
↓	π	Lower: Unshifted

**Boldface** indicates that the character is printed on the keytop.

When no character is shown in the lower area of the keytop, one of the following is true:

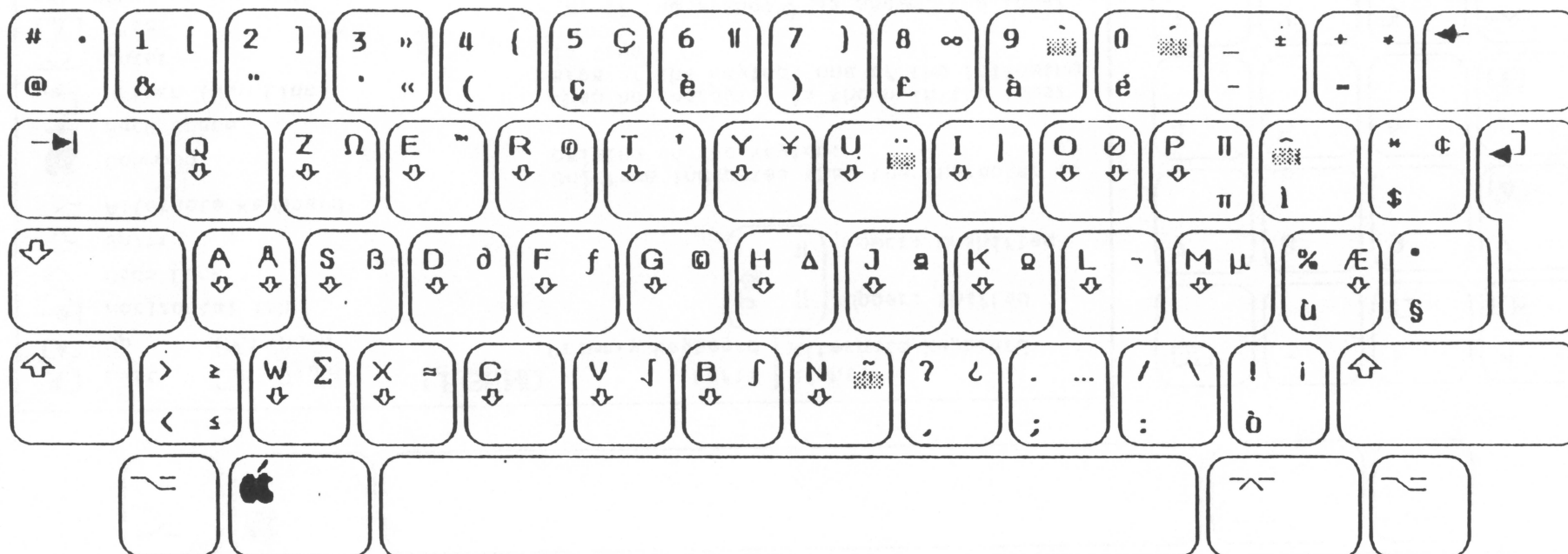
- If the symbol is shown, the lower character is the lower case equivalent of the upper character.
- If the symbol is not shown, the lower character is the same as the upper character.





# Italian Keyboard Layout

TN / 12 Nov, 82



Left Right  
Up Down

Horizontal Tab

Caps Lock

Shift

Alternate Keyboard

Command

Back Space

Return (New Line)

Enter

Clear

Character has lower case and is affected by Caps Lock.

Dead key, no character is generated until next key is pressed.



## LEGEND

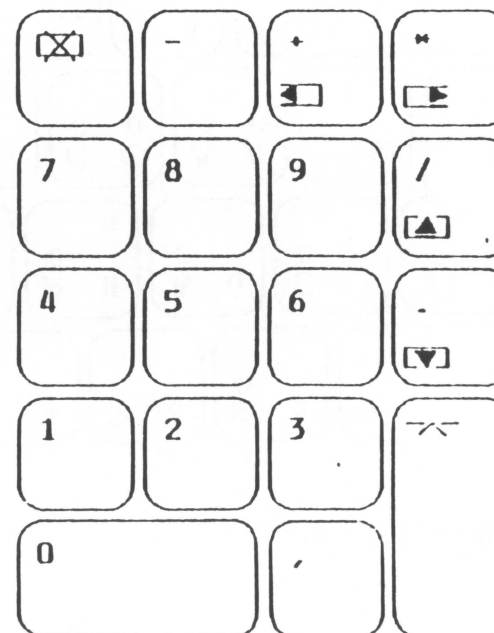
Left: Right:  
Primary keyboard Alternate keyboard

Upper: Shifted  
Lower: Unshifted

Boldface indicates that the character is printed on the keytop.

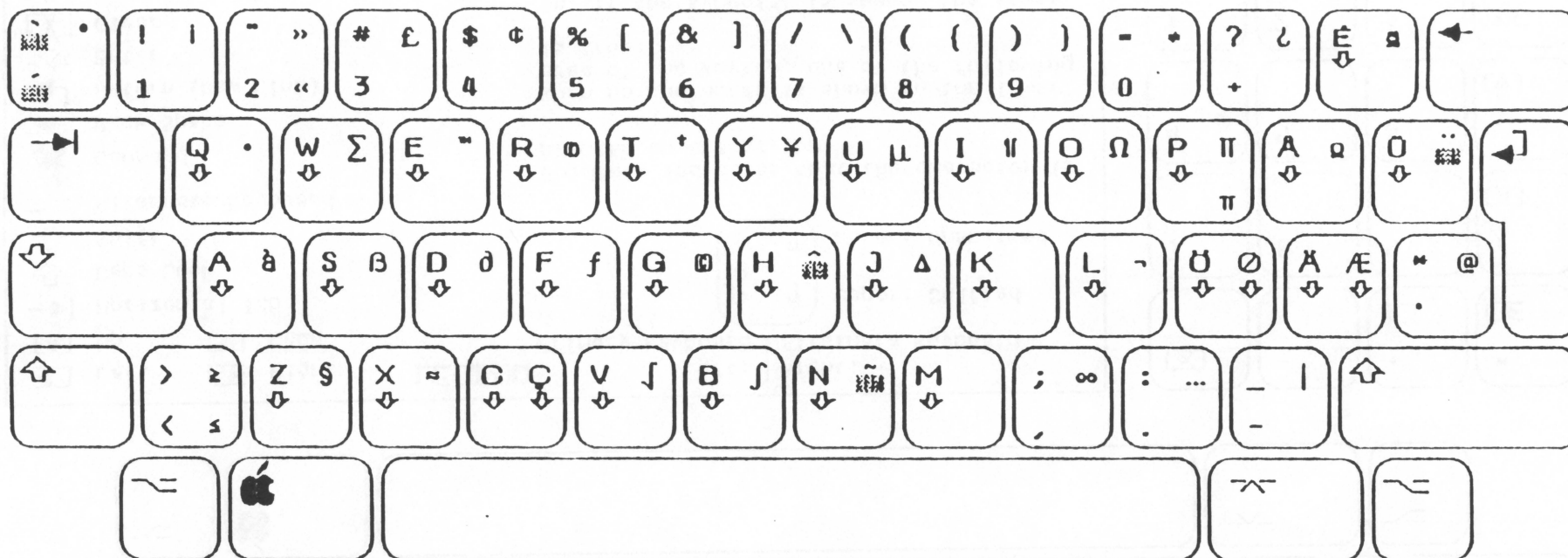
When no character is shown in the lower area of the keytop, one of the following is true:

- If the symbol  is shown, the lower character is the lower case equivalent of the upper character.
- If the symbol  is not shown, the lower character is the same as the upper character.



# Swedish Keyboard Layout

TN / 8 Nov 82



Left

Right

Up

Down

Horizontal Tab

Caps Lock

Shift

Alternate Keyboard

Command

Back Space

Return (New Line)

Enter

Clear

Character has lower case and is affected by Caps Lock.

Dead key, no character is generated until next key is pressed.

### LEGEND

Left:

Primary keyboard

Right:

Alternate keyboard

**P** **Π** Upper: Shifted

**Π** Lower: Unshifted

**Boldface** indicates that the character is printed on the keytop.

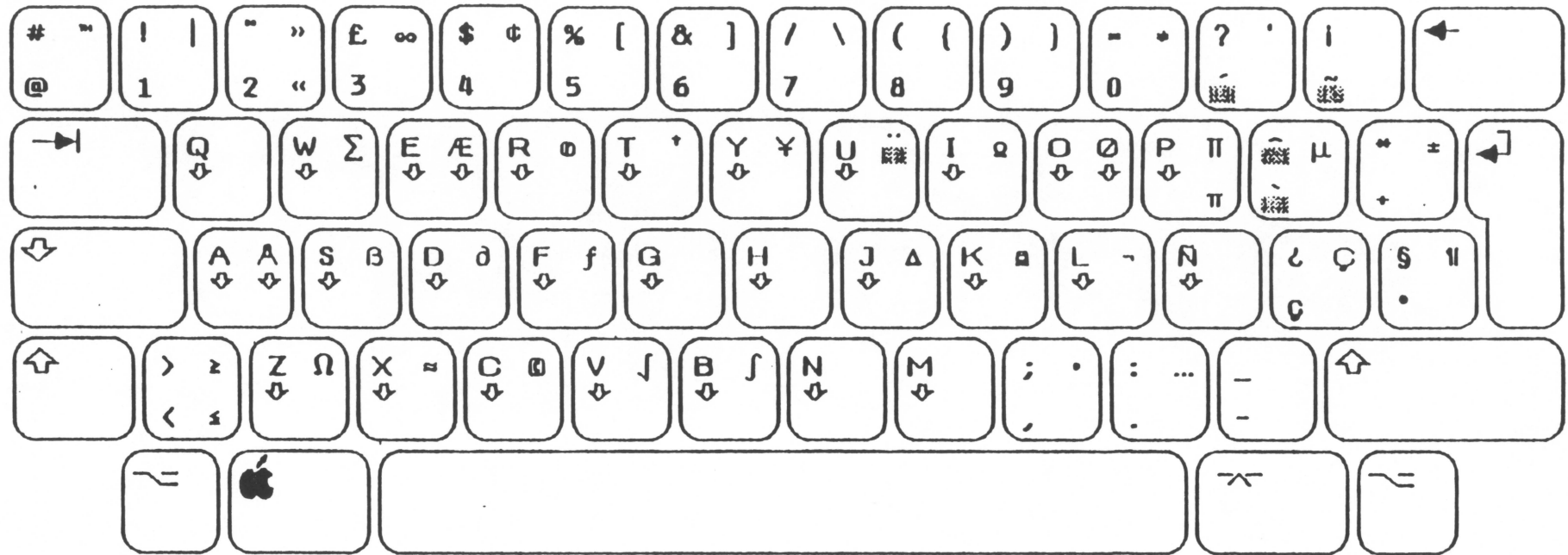
When no character is shown in the lower area of the keytop, one of the following is true:

- If the symbol is shown, the lower character is the lower case equivalent of the upper character.
- If the symbol is not shown, the lower character is the same as the upper character.



# Spanish Keyboard Layout

TN / 8 Nov, 82



- Left
- Right
- Up
- Down
- Horizontal Tab
- Caps Lock
- Shift
- Alternate Keyboard



Command



Back Space



Return (New Line)



Enter



Clear



Character has lower case and is affected by Caps Lock.



Dead key, no character is generated until next key is pressed.

## LEGEND

Left: Primary keyboard Right: Alternate keyboard



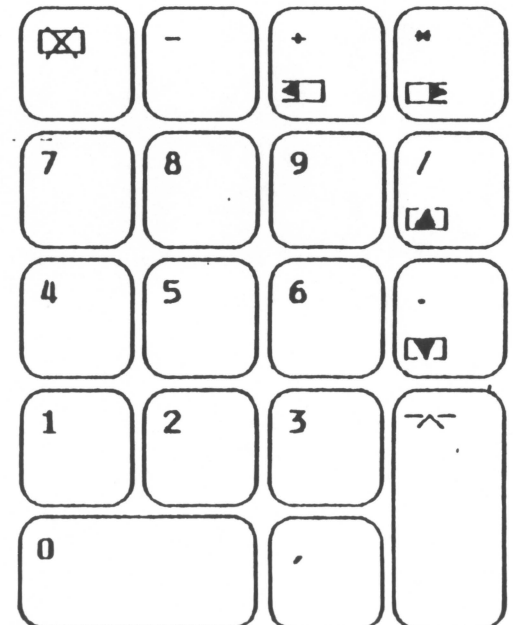
Upper: Shifted

Lower: Unshifted

**Boldface** indicates that the character is printed on the keytop.

When no character is shown in the lower area of the keytop, one of the following is true:

- a. If the symbol is shown, the lower character is the lower case equivalent of the upper character.
- b. If the symbol is not shown, the lower character is the same as the upper character.





Appendix I. Video Assembly Schematic

---

Paula Nunez will supply camera ready copy.

Appendix A: Formal Notation

This section will discuss the formal notation used in this book.

## Appendix J. Radio and Television Interference

---

The Lisa generates and uses radio frequency energy. If the Lisa is not installed and used properly (that is, in strict accordance with these instructions), it may cause interference to radio and television reception.

This equipment has been tested and complies with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC rules. These rules are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that the interference will not occur in a particular installation, especially if a "rabbit ear" TV antenna is used. (A rabbit ear antenna is the telescoping rod antenna usually contained on TV receivers.)

You can determine whether your computer is causing interference by turning it off. If the interference stops, it was probably caused by the computer or its peripherals. To further isolate the problem:

- o Disconnect the peripheral devices and their I/O cables one at a time. If the interference stops, it is caused by either the peripheral or its I/O cable. These devices usually require shielded I/O cables. For Apple peripherals, you can obtain the proper shielded cable from your dealer. For non-Apple peripherals, contact the manufacturer or your dealer for assistance.

If your computer does cause interference to radio or television reception, you can try to correct the interference by using one or more of the following measures:

- o Turn the TV or radio antenna until the interference stops.
- o Move the computer to one side or the other of the TV or radio.
- o Move the computer farther away from the TV or radio.
- o Plug the computer into an outlet that is on a different circuit from the TV or radio. That is,

make certain the computer and the TV or radio are on circuits controlled by different circuit breakers or fuses.

- o Consider installing a rooftop TV antenna with coaxial cable lead-in between the antenna and the TV.

If necessary, you should consult your dealer or an experienced radio/television technician for additional suggestions. You may find helpful the following booklet, prepared by the Federal Communications Commission: How to Identify and Resolve Radio-TV Interference Problems. This booklet is available from the U.S. Government Printing Office, Washington, D.C. 20402, Stock Number 004-000-003454.









July 2, 1985

Dear Developer,

Enclosed is your copy of the final **Software Supplement Update** (dated May 1985). The enclosed documentation includes:

<b>Cover Sheet</b>	2	pages
<b>About the "May 1985" Software Supplement</b>	54	
<b>Macintosh Technical Documentation Order Form</b>	1	
<b>Switcher (Beta Draft)</b>	16	
<b>A Software Developer's Guide to Switcher</b>	3	
<b>Driver Bug in Pre-Release MacWorks XL</b>	2	
<b>Technical Note #0: About Macintosh Technical Notes</b>	2	
<b>Technical Note #16: MacWorks XL</b>	3	
<b>Technical Note #32: Reserved Resource Types</b>	1	
<b>Macintosh Update for End-Users</b>	10	
<b>Trap List</b>	(1 cover page +) 16	
<b>Welcome To MAUG™</b>	3	
<b>FreeTerm</b>	6	
<b>ResEdit: A Macintosh Resource Editor</b>	8	
<b>AppleTalk Information</b>	3	
<b>AppleTalk Peek</b>	9	
<b>AppleTalk Poke</b>	11	
<b>MacinTalk 1.1</b>	17	
<b>The WriteIn Window</b>	2	
<b>Low Memory in Alphabetical Order</b>	2	
<b>Low Memory in Numerical Order</b>	2	
<b>Putting Together A Macintosh Application</b>	23	
<b>The Macintosh Hardware</b>	37	
<b>The Printing Manager</b>	31	
<b>Examples of Printing Manager Usage</b>	2	
<b>Some Words of Wisdom About Using QuickDraw While Printing</b>	1	
<b>The March 1985 ImageWriter: Programmers' Notes</b>	5	
<b>Optimizing Code For The LaserWriter</b>	6	
<b>Future Macintosh Architectures</b>	6	
<b>Finders and Foreign Drives</b>	1	
<b>Life After Font/DA Mover--How To Make Sure Your Desk Accessory Still Works</b>	2	
<b>SANELib V1.2</b>	1	
<b>SANE Numeric Scanner and Formatter</b>	4	
<b>DIALOG CREATOR Instructions</b>	8	
<b>Fedit: A File And Disk Editor</b>	22	

The following enclosed documents are printed on smaller pages so they are wrapped separately:

**Macintosh REdit: A Macintosh Resource Editor** 20

**Revision to Workshop User's Guide for the Lisa**

<b>Cover Sheet</b>	1
<b>Chapter 2: The File Manager</b>	(1 cover page +) 31
<b>Chapter 4: The Editor</b>	(1 cover page +) 25
<b>Index</b>	15

The following documents were distributed with previous Supplement Updates but are still relevant; they are *not* enclosed:

**Commented Call List**  
**Latest "Post-3.0" Lisa Pascal Compiler Enhancements**  
**Macintosh PasLib Release 0.7**  
**MacWorks XL User Manual**  
**The MacDB Debugger**  
**The MacsBug Debuggers**

The thirteen disks enclosed with this Supplement Update are labelled:

**Workshop 3.9 Update Disk 1**  
**Workshop 3.9 Update Disk 2**  
**5/85 Workshop Supplement 1**  
**5/85 Workshop Supplement 2**  
**5/85 Workshop Supplement 3**  
**5/85 Examples 1**  
**5/85 Examples 2**  
**5/85 MacStuff 1**  
**5/85 MacStuff 2**  
**5/85 MacStuff 3**  
**5/85 MacStuff 4**  
**5/85 Mac Build Disk**  
**MacWorks XL 3.0**

If you have questions about missing or damaged materials (disks or documentation), please contact our mailing facility at:

Apple Computer Mailing Facility/Milestone Group  
467 Saratoga Avenue, Suite 621  
San Jose, CA 95129

Customer Service:  
(408) 988-6009  
9:00 A.M. - 4:00 P.M., Pacific Time

# About the "May 1985" Software Supplement

May 3F, 1985

## Table of Contents

Table of Contents	1
Future Distributions	2
Disks in the Supplement	3
Documentation Accompanying the Software Supplement	4
** Development Using the Lisa Pascal Workshop	7
** Workshop 3.9 Update Disks	8
** Installing the Workshop Supplement disks	9
** Contents of the Workshop Supplement Disks	10
** Files on 5/85 Workshop Supplement 1	10
** Files on 5/85 Workshop Supplement 2	11
** Files on 5/85 Workshop Supplement 3	12
** Workshop Pascal Interfaces	14
** Text Files	14
* Object Files	15
** Changes to Pascal Interfaces	17
Assembler Equates	19
Equate Files	19
Changes to Assembler Equate Files	20
** Workshop 3.9--New Features	21
** Workshop Shell	21
** Pascal Compiler and Code Generator	21
** Linker	21
** SANELib	22
** MacCom	22
** RMaker	23
** Workshop Editor	24
** Assembler	25
** Resource File Builder (RFB)	29
* Example Programs and Exec Files	31
* Files on 5/85 Examples 1	32
* Files on 5/85 Examples 2	33
* MacWorks™ XL	37
Mac Build Disk	38
MacStuff Disks	39
Files on 5/85 MacStuff 1	39
Files on 5/85 MacStuff 2	41
Files on 5/85 MacStuff 3	43
Files on 5/85 MacStuff 4	44
FixMath and Graf3D	47
SANE	47
RAM-Based Serial Drivers	48
Debuggers	49
Macintosh Product Availability List	51
Macintosh Pascal	51
* MacApp	52
Smalltalk	53
Addresses	54

\* = Mainly of interest to Lisa users  
\*\* = Only of interest to Lisa users



## Future Distributions

This is the final update of the Software Supplement. For the last year we have periodically been distributing updates to the Supplement to all buyers. Inside Macintosh and the software interfaces and tools were frequently changing and we wanted everyone to have the most up-to-date versions. The software in this version of the Supplement corresponds to the final version of Inside Macintosh. As promised, Supplement purchasers will receive a copy of the final bookstore version of Inside Macintosh when it is available this Fall.

We may periodically offer new development tools and utilities as products which can be ordered independently. When new products are available, we will let Supplement buyers know how to obtain them. In addition, many of these new tools and utilities will be available from MAUG<sup>TM</sup> via Compuserve. Anyone with a modem can use download copies. FreeTerm, a communication program which will allow you to do so, has been included in this Supplement. More information on these on-line services is included in the enclosed documents titled **Welcome To MAUG<sup>TM</sup>** and **FreeTerm**.

## Disks in the Supplement

The Macintosh Software Supplement is a package of tools, libraries, and examples to help you develop Macintosh software. This final update to the Supplement consists of the following thirteen disks:

**Workshop 3.9 Update Disk 1**  
**Workshop 3.9 Update Disk 2**

Lisa Workshop 3.0 formatted disks.  
Update Workshop 3.0 to Workshop 3.9.

**5/85 Workshop Supplement 1**  
**5/85 Workshop Supplement 2**  
**5/85 Workshop Supplement 3**

Lisa Workshop 2.0 formatted disks;  
can be used with any version of the Lisa Workshop.

**5/85 Examples 1**  
**5/85 Examples 2**

Macintosh formatted disks. Contain source text of  
example programs. Need Lisa Workshop to build.

**5/85 MacStuff 1**  
**5/85 MacStuff 2**  
**5/85 MacStuff 3**  
**5/85 MacStuff 4**

Macintosh formatted disks.  
Contain tools, utilities and examples which can be  
used on any Macintosh.

**5/85 Mac Build Disk**

Macintosh formatted disk. Contains standard system  
software. Should be used when building a software  
product disk.

**MacWorks XL (version) 3.0**

Special format--used on a Lisa or Macintosh XL to  
run Macintosh applications.

This package contains new versions of *all* the above disks; we recommend putting aside any older versions of the Supplement and using these disks exclusively (if you are using Pascal Workshop 2.0 you will need some files from the February 1985 Supplement disks).

# Documentation Accompanying the Software Supplement

The documents listed below have been included with this Software Supplement update. Those marked with a "\*" are primarily of interest to Lisa users.

The **Cover Sheet** lists the documents included with this Supplement and the number of pages in each. It also lists the disks in this Supplement.

**About the "May 1985" Software Supplement**, the document you are reading now, describes the contents of this Software Supplement and how to use the various pieces.

The **Macintosh Technical Documentation Order Form** should be used to order most Macintosh technical documentation from our mailing house.

The **Switcher (Beta Draft)** describes Switcher 3.0 (a pre-release version) from a user's point-of-view. A more complete version of this manual will be available from MAUG™ (via Compuserve) and from Apple dealers when Switcher is released.

A **Software Developer's Guide to Switcher** describes the Switcher from an application writer's point-of-view. Updates to this document will be available from MAUG™ (via Compuserve).

**Driver Bug in Pre-Release MacWorks XL** describes a serious bug in MacWorks XL (which has been corrected in MacWorks XL 3.0) and how an application can patch MacWorks to avoid problems.

**Technical Note #0: About Macintosh Technical Notes** describes our Technical Notes service and how to subscribe. It also lists the Technical Notes which have been published to date.

**Technical Note #16: MacWorks XL** describes some features of MacWorks XL. Note that MacWorks XL version 3.0 has been released since this document was written; see **Driver Bug in Pre-Release MacWorks XL** for more information.

**Technical Note #32: Reserved Resource Types** lists the resource type names reserved for use by Apple. Refer to this list before naming any custom resource types.

**Macintosh Update for End-Users** describes Finder 4.1, Choose Printer, the Font/Desk Accessory Mover and the corresponding System file from a user's point-of-view.

The **Trap List** is a list of traps including: the trap or routine name as it is described from Pascal, the trap word, the section in Inside Macintosh where it is discussed, how the routine affects the heap, and a list of what other traps are called by the routine.

**Welcome To MAUG™** describes how to use MAUG™, the user's group on Compuserve through which future Macintosh development tools and utilities will be distributed.

The **FreeTerm** document describes FreeTerm, the simple terminal emulator tool which can be used to access Compuserve.

**ResEdit: A Macintosh Resource Editor** describes release 0.5 of the Macintosh resource editor ResEdit.

The **AppleTalk Information** document contains some important information for anyone writing an application which uses the AppleTalk network. It contains a questionnaire which such developers should complete and return to Apple.



**AppleTalk Peek** describes the Peek utility which can be used to monitor packet traffic on an AppleTalk network.

**AppleTalk Poke** describes the Poke utility which can be used to create or edit packets and send them out on AppleTalk.

**MacinTalk 1.1** describes the MacinTalk speech synthesizer which can be used by Macintosh applications. MacinTalk software is included in this Supplement.

- \* **The Writeln Window** describes the unit WritelnWindow which can be used for debugging Macintosh programs written in Lisa Pascal.

**Low Memory in Alphabetical Order** lists all the low memory equates in alphabetical order.

**Low Memory in Numerical Order** lists all the low memory equates in numerical order.

- \* **Putting Together A Macintosh Application** describes how to build a Macintosh application from the Lisa Pascal Workshop. This sixth draft, dated 5/5/85, is the final revision of this manual. It is no longer a part of Inside Macintosh.

**The Macintosh Hardware** is the hardware chapter for Inside Macintosh which has not previously been released. This second draft is dated 2/13/85.

**The Printing Manager** is the chapter from Inside Macintosh describing how applications can print. This second draft, date 3/27/85, is significantly improved from the first draft available previously.

**Examples of Printing Manager Usage** provides examples of how an application could call the printing manager.

**Some Words of Wisdom About Using QuickDraw While Printing**, as the title suggests, provides some tips on how to best use QuickDraw when printing.

**The March 1985 ImageWriter: Programmers' Notes** describes how to print using the ImageWriter driver released in March 1985 and included on the 5/85 Mac Build Disk.

**Optimizing Code For The LaserWriter** describes how to make your application better at printing to the LaserWriter printer.

**Future Macintosh Architectures** lists a number of guidelines that should help ensure that your software will continue to run on any future Apple Macintosh hardware. All developers should read this document; those writing in assembly language should read this document especially carefully.

**Finders and Foreign Drives** describes the interaction between Finder 4.1 and non-Apple disk drives.

**Life After Font/DA Mover--How To Make Sure Your Desk Accessory Still Works** describes the Font/Desk Accessory Mover from the point of view of a desk accessory developer.

- \* The document **SANELib V1.2** describes how to access SANE (Standard Apple Numeric Environment) floating point libraries from the Lisa Pascal Workshop 3.9.

**SANE Numeric Scanner and Formatter** describes SANE's utility routines for conversion between Decimal records and ASCII strings and how to use these routines from various development systems.

**DIALOG CREATOR Instructions** describes the Dialog Creator tool which can be used to easily create resource definition files for dialogs and alerts.

**Fedit: A File And Disk Editor** describes Fedit, a "shareware" utility program which allows somewhat technical users to access Macintosh disks at a low level.

**Macintosh REdit: A Macintosh Resource Editor**, a booklet which is wrapped separately, describes the tools REdit and Localizer, which can be used for modifying applications for international markets. REdit can also be used as a general purpose resource editor.

- \* **Revision to Workshop User's Guide for the Lisa** is also wrapped separately. It updates sections of the *Lisa Workshop 3.0 Users's Guide* (note that the Workshop 3.9 Update materials in this Supplement are more recent than these sections). The sections include a **Cover Sheet, Chapter 2: The File Manager, Chapter 4: The Editor, and the Index.**

The following documents were distributed with previous updates to the Software Supplements but are still relevant. *They will only be included in this package if this is the first time you have received the Software Supplement.*

The **Commented Call List** document can be used as a quick reference to the Pascal Interfaces for the Macintosh. It lists the Procedure and Function calls for most of the Toolbox and OS managers (notable exclusions include Quickdraw and the Print Manager) grouped by manager. Within each manager the calls are ordered from the most frequently used calls to dangerous or obscure calls. The calls are accompanied by brief usage notes.

- \* The memo **Latest "Post-3.0" Lisa Pascal Compiler Enhancements**, dated February 8, 1985, describes the Pascal compiler for Workshop 3.9 which is included in this Supplement. It includes details on the use of SANE from Pascal. Appendix A is no longer accurate and has been removed from new copies of the document.
- \* **Macintosh PasLib Release V.0.7** describes the latest PasLib library for use from Lisa Pascal.
- \* The **MacWorks XL User Manual** describes MacWorks XL from a user's perspective. The information in this manual applies to MacWorks XL and MacWorks XL 3.0

The **MacDB Debugger**, chapter 6 of the Macintosh 68000 Development System User's Manual. This describes MacDB, the "two-Mac debugger".

The **MacsBug Debuggers**, chapter 7 of the Macintosh 68000 Development System User's Manual. This describes the MacsBug family of debuggers. Updates to this manual can be found in the **Debuggers** section of **About the "May 1985" Software Supplement**.

## Development Using the Lisa Pascal Workshop

In order to develop Macintosh software using the Lisa Workshop you need a Lisa 2/5 or 2/10 (also known as Macintosh XL) with at least a full megabyte of RAM, this Supplement, and the Lisa Pascal Workshop, version 2.0 or 3.0. You must install the Pascal Workshop on a hard disk before attempting to install the Supplement.

If you want to have both the Lisa Office System and the Workshop with Supplement, you'll find that one five megabyte ProFile isn't enough. You'll need a Lisa 2/10 (Macintosh XL), a ten megabyte ProFile, or separate five megabyte ProFiles. Of course, you don't need the Lisa Office System to do Macintosh development.

*Pascal Workshop 2.0 users* can continue to use the 2.0only/ files included in the 2/15/85 Supplement. They can use all files on the **Workshop Supplement** disks that are not prefixed with 3.9only/. They cannot use the **Workshop 3.9 Update** disks. Although developers can continue to use Workshop 2.0 we *strongly* recommend upgrading to Workshop 3.0.

Pascal Workshop 3.0 users can update their system to Pascal Workshop 3.9 by performing the Workshop 3.9 Update procedure described on the following page. They can then use all of the files on the **Workshop Supplement** disks (except the few that are prefixed with 2.0only/).

Pascal Workshop 3.0 is compatible with all versions of the Lisa 7/7 Office System and allows the hard disk to be shared with a MacWorks volume. Other improvements over Workshop 2.0 include:

- improved Editor
- compiler enhancements
- many new Workshop utilities
- better performance
- hierarchical file system (subdirectories)
- improved MacCom and RMaker
- improved access to SANE floating point
- capability for additional enhancements by installing Workshop 3.9  
(you'll need 3.0 to install the 3.9 update,  
which is included in this Supplement)

Users of Lisa Pascal 2.0 can receive a new copy of Pascal Workshop 3.0 (the full product, not just an upgrade kit) by sending their original Pascal Workshop 2.0 master disk (Pascal 1) and a check for \$150 (plus local sales tax for California residents) to:

Apple Computer, Inc.  
3.0 Upgrade  
467 Saratoga Ave. Suite 621  
San Jose, CA 95129

(408) 988-6009

You should make a copy of your Pascal 1 disk before sending in the original. Please allow approximately 4 weeks for delivery.

**This upgrade offer expires on August 31, 1985.** After that date new copies of the Lisa Pascal Workshop 3.0 (Apple part #A6D0301) will continue to be available through regular Apple channels.

## Workshop 3.9 Update Disks

The Workshop 3.9 Update is an update to the Pascal Workshop 3.0 release. It contains the latest, most up-to-date Workshop tools, including a new "Post-3.0" Pascal compiler with supporting libraries, the latest SANE libraries, and new versions of the Workshop Shell, the linker, the editor, the assembler, RMaker, and MacCom. The only Workshop 3.0 users who should *not* install the entire update are those who are still using the "Old World" SANE floating point described in the February Supplement.

The Workshop 3.9 Update consists of two disks, **Workshop 3.9 Update Disk 1** and **Workshop 3.9 Update Disk 2**. These disks have been formatted using the Lisa Pascal Workshop version 3.0, so *they cannot be read from a Lisa running Workshop 2.0 or from a Macintosh*. However, there is nothing on these disks which is useful to anyone who does not have Pascal Workshop 3.0. The disks contain the following files:

### Files on Lisa Workshop 3.0 disk Workshop 3.9 Update Disk 1:

Assembler.obj  
Code.obj  
Editor.obj  
IOSPasLib.obj  
IUManager.obj  
Linker.obj  
Mac.boot  
MacCom.obj  
StartUpdate.text  
tmp/ContinueOrAbort.text  
tmp/DoUpdate.text  
tmp/GetDisk.text  
tmp/YesNoFunc.text

### Files on Lisa Workshop 3.0 disk Workshop 3.9 Update Disk 2:

intrfc/SANELib.text  
obj/SANELib.obj  
obj/SANELibAsm.obj  
OSErrs.Err  
Pascal.obj  
PasErrs.Err  
RMaker.obj  
Shell.Workshop

### **How to install the Workshop 3.9 Update:**

An automatic exec update procedure is provided to facilitate the installation of the new software. It should work on all configurations of Workshop 3.0, even if Lisa 7/7 is installed on the same disk. Before starting, make sure that you have at least 1000 blocks free on the hard disk that you will be updating (this is recommended for any work with the Workshop). If necessary, you may be able to free up some space by booting from another Workshop profile and Scavenging or booting from **Pascal 3.0 disk 1** and choosing "*Repair*"). Insert the **Workshop 3.9 Update disk 1** (note that the Workshop will not accept write-protected disks) and then invoke the "StartUpdate.text" exec file by using the run command as follows:

**R<-lower-StartUpdate**

The update exec files will lead you through the rest of the update procedure (including allowing you to choose which hard disk to update, prompting you to install **Workshop 3.9 Update Disk 2** when necessary, deleting the tmp/ files it uses, and finally asking you to reboot).



## Installing the Workshop Supplement Disks

The Workshop Supplement disks (formerly called "MacSupplement" disks) contain interfaces, equates, exec files, etc. that can be used with the Lisa Pascal Workshop. Workshop 3.0 users should update to Workshop 3.9 and reboot before installing the Workshop Supplement disks.

Read through the descriptions of the Workshop Supplement files on the following pages and choose the files you need. To install the Workshop Supplement disks onto your hard disk, start the Workshop, then insert each of the disks and use the **Backup** command to copy the desired files from the Workshop Supplement disk to the hard disk (note that this will automatically replace all files with the same names as Supplement files, so you may want to look at the list of files on the Supplement disks before copying them with **Backup**). If your hard disk is the default volume, the following command will copy all the files from the currently inserted 3 1/2" disk to your hard disk:

**B-lower-=,\$**

Note that the Workshop will not accept write-protected disks.

This Supplement supports Workshop versions 2.0 and 3.9. The disks are provided in 2.0 format, which is readable by both versions. Most of the files provided with the Supplement are usable by both Workshop versions; the only files which are specific to one of the two versions are prefixed by "2.0only/" or "3.9only/". After you copy all desired files use the **Rename** command to strip the prefix from the files that have one. The command would look like this:

**R2.0only/=,=**

or

**R3.9only/=,=**

Note that some of these files are replacements for files you already have, so the **Backup** command will ask if you want to delete the old ones before renaming. You should answer **yes** to this question.

Workshop 3.9 users will not need files prefixed with " 2.0only/" on their hard disk.

Workshop 2.0 users will not need files prefixed with " 3.9only/" on their hard disk (this includes *all* files on the **5/85 Workshop Supplement 3** disk).

Workshop 2.0 users should note that they still need the 2.0only/ files from the February 1985 Software Supplement. Workshop 3.0 users who are using the "Old World" SANE will also need some of the " 2.0only/" files found in the February Supplement. New owners of the Supplement should use Workshop 3.0 (and upgrade to Workshop 3.9).

If you need more room on your hard disk, you can delete some files. Appendix I of the Pascal 3.0 Reference Manual (labelled "Lisa Language" on the spine) lists the files that come with the Workshop and indicates the purpose of each. Those marked E,F, or G are not needed for Macintosh development (except for `sys2Lib.Obj`, which is needed to run Preferences). Disks 7, 8 and 9 of Pascal 3.0 are completely optional for Macintosh development. In addition, if you're not doing any assembly language development, you can delete `Assembler.Obj` and all files which begin with `TLAsm/` (however, these are needed to build the assembly portion of some sample programs).

## Contents of the Workshop Supplement disks

### Files on Lisa Workshop disk 5/85 Workshop Supplement 1:

intrfc/ABPasIntf.text  
intrfc/FixMath.text  
intrfc/Graf3D.text  
intrfc/MacPrint.text  
intrfc/MemTypes.text  
intrfc/OSIntf.text  
intrfc/PackIntf.text  
intrfc/PasLibIntf.text  
intrfc/QuickDraw.text  
intrfc/QuickDraw2.text  
intrfc/SpeechIntf.text  
intrfc/ToolIntf.text  
intrfc/WritelnWindow.text  
obj/ABPasCalls.obj  
obj/ABPasIntf.obj  
obj/FixAsm.obj  
obj/FixMath.obj  
obj/Graf3D.obj  
obj/Graf3DAsm.obj  
obj/MacPrint.obj  
obj/MemTypes.obj  
obj/OSIntf.obj  
obj/OSTraps.obj  
obj/PackIntf.obj  
obj/PackTraps.obj  
obj/PasInit.obj  
obj/PasLib.obj  
obj/PasLibAsm.obj  
obj/PasLibIntf.obj  
obj/PrLink.obj  
obj/PrScreen.obj  
obj/QuickDraw.obj  
obj/RTLib.obj  
obj/SpeechAsm.obj  
obj/SpeechIntf.obj  
obj/ToolIntf.obj  
obj/ToolTraps.obj  
obj/WritelnWindow.obj

The **5/85 Workshop Supplement 1** disk contains many `intrfc/` files, which are human readable text versions of the Pascal interfaces the Macintosh Toolbox, OS, Packages, and QuickDraw units; and `obj/` files, which are the object files for the Pascal interfaces (used for compiling and linking). For more information on these files see the section below titled **Workshop Pascal Interfaces**. `PasLib` and `WritelnWindow` are discussed in separate documents.

The `intrfc/` and `obj/` files (except for `WritelnWindow`) are part of the Pascal Interface version 1.1 (the February Supplement contained beta versions of the 1.1 interfaces); these are the final versions, corresponding to the forthcoming published edition of Inside Macintosh.

### Files on Lisa Workshop disk 5/85 Workshop Supplement 2:

2.0only/Convert/Mac2Lisa1.text  
2.0only/Convert/Mac2Lisa2.text  
2.0only/Convert/TextConvert.obj  
3.9only/RMaker7.14a.obj  
ATalk/ABPackage.obj  
ATalk/ABPackageR.text  
intrfc/WritelnWindow2.text  
Serial/Async/Mac.obj  
Serial/Async/MacXL.obj  
Serial/AsyncR.text  
source/RFB.text  
TLAsm/ATalkEqu.text  
TLAsm/FSEqu.text  
TLAsm/HardwareEqu.text  
TLAsm/PackMacs.text  
TLAsm/PrEqu.text  
TLAsm/QuickEqu.text  
TLAsm/QuickTraps.text  
TLAsm/SaneMacs.text  
TLAsm/SysEqu.text  
TLAsm/SysErr.text  
TLAsm/SysTraps.text  
TLAsm/ToolEqu.text  
TLAsm/ToolTraps.text

The **5/85 Workshop Supplement 2** disk contains the TLAsm/ files, the version 1.1 equate files for the Lisa assembler. The TLAsm/ files define macros and symbols for assembly language programs and are equivalent to the MDS Equate files on the **5/85 MacStuff 4** disk. For more information, see the **Assembly Equates Information** section.

The disk also contains a number of specialized files. The 2.0only/Convert/ files convert Macintosh text files to Lisa Workshop 2.0 text files (MacCom supports this for 3.0 users). This is described in the **Example Programs and Exec Files** section. The file 3.9only/RMaker7.14a.obj is only needed by a few users; see the **RMaker** section of this document. The ATalk/ files are described in the document **AppleTalk Information**, included with this Supplement. The file intrfc/WritelnWindow2.text contains the source for the WritelnWindow unit (it should be used with intrfc/WritelnWindow.text). The Serial/ files are needed to include the latest RAM-based serial driver in a resource definition file (see the **RAM-Based Serial Drivers** section of this document and the file serial/AsyncR.text for more details). The file source/RFB.text is source code to the Resource File Builder, described in the **Resource File Builder (RFB)** section of this document.

### Files on Lisa Workshop disk 5/85 Workshop Supplement 3:

3.9only/convert/Mac2Lisa.text \*

3.9only/DumpObj.obj

3.9only/example/ExampleList.text

3.9only/example/Exec.text \*

3.9only/example/ExecAll.text

3.9only/example/ExecAll2.text

3.9only/example/Graf3DLink.text

3.9only/example/MaxLink.text

3.9only/example/MinLink.text

3.9only/example/PrintLink.text

3.9only/example/SANELink.text

3.9only/example/SpeechLink.text

3.9only/example/VanillaExec.text

3.9only/example/WritelnLink.text

3.9only/Lisa/SANELib.obj

3.9only/Lisa/SANELib.text

3.9only/Lisa/SANELibAsm.obj

3.9only/ProcNames.Help.text

3.9only/ProcNames.obj

3.9only/REdit.obj

3.9only/REdit/Userguide.text

3.9only/RFB.obj

3.9only/RFB/exec.text

3.9only/RMaker.obj \*

3.9only/ShowInterface.Help.text

3.9only/ShowInterface.obj

3.9only/SXref.Assembly.text

3.9only/XRef.Help.text

3.9only/XRef.obj

\* = important; of interest to most developers

The **5/85 Workshop Supplement 3** disk contains files for use with Workshop 3.9. Workshop 2.0 owners should note that they will not need any files from this disk. Users of Workshop 3.9 should strip off the 3.9only/ prefix either when copying the files from the disk (using the file manager backup command `B3.9only/=,=`) or by renaming (with the command `R3.9only/=,=`). Note that only the files marked with a \* will be used by most developers; the others are useful in certain situations; you may wish to only copy those you have a need for onto your hard disk.

This disk contains the latest versions of the following optional Pascal utilities: DumpObj (v. 3.2), ProcNames (v4. 31), ShowInterface (v. 1.5), and XRef (v. 4.39). These tools now know how to handle the latest Pascal syntax and the results of partial links (partial linking is described later in this document in the **Linker** subsection of the **Workshop 3.9--New Features** section). DumpObj has a new "Entry points only" option. These versions of these tools should only be used after updating to Workshop 3.9. They replace the optional Workshop utilities of the same name found on disk 7 of Lisa Pascal Workshop 3.0. The files ProcNames.Help.text, ShowInterface.Help.text, and XRef.Help.text are provided on this disk for your convenience.

The file 3.9only/convert/Mac2Lisa.text is an exec file which uses MacCom to convert Macintosh text files to Lisa Workshop text files. The 3.9only/example/ files are exec files and inputs to the exec files. See the **Example Programs and Exec Files** section of this document for more information about all of these.



The three 3.0only/Lisa/ files are only required if you plan to write and execute programs involving floating point numbers under the *Lisa Operating System* (not just executing them on the Macintosh). See the document **SANELib V1.2** for more information.

3.9only/Redit.obj is a resource editor that's documented in the file Redit/Userguide.text (additional resource editors that run on the Macintosh can be found on the **5/85 MacStuff 1** disk).

The files 3.9only/RFB.obj and 3.9only/RFB.exec.text are described in the **Resource File Builder (RFB)** section of this document.

The file 3.9only/RMaker.obj is version 7.14b of RMaker, described in the **RMaker** section of this document.

The file 3.9only/SXref.Assembly.text replaces the file SXref.Assembly.text found on disk 7 of Lisa Pascal Workshop 3.0. It is used by the Workshop utility SXref (which has not changed) when formatting assembly language source.

For additional information on using the Supplement with the Pascal Workshop, see the 5/5/85 update to **Putting Together a Macintosh Application** which is included with this Supplement.

## Workshop Pascal Interfaces

A new release of the interface files needed for Lisa Pascal development for the Macintosh is included in this supplement. This release is version 1.1 (the February Supplement contained a beta-release of version 1.1). Not too many changes have been made since the February release. These files should be the basis for all future Macintosh development in Pascal.

### Text Files

These files are for human consumption. They are the interface portions of the various libraries and include the relevant constants, types, and routine definitions.

<code>intrfc/ABPasIntf.text</code>	AppleTalk Pascal interface
<code>intrfc/FixMath.text</code>	Fixed point math
<code>intrfc/Graf3D.text</code>	Three-dimensional graphics routines layered on top of QuickDraw. Use with FixMath.
<code>intrfc/MacPrint.text</code>	Device independent printing
<code>intrfc/MemTypes.text</code>	Common types
<code>intrfc/OSIntf.text</code>	Operating system routines (Memory Mgr, File Mgr, Sound Driver, RAM serial driver, ...)
<code>intrfc/PackIntf.text</code>	Packages (Standard File, International, Binary-Decimal conversion, Disk initialization, ...)
<code>intrfc/PasLibIntf.text</code>	PasLib (non built-in) functions dealing with the heap and Writeln redirection.
<code>intrfc/QuickDraw.text</code>	Graphics routines
<code>intrfc/QuickDraw2.text</code>	Implementation stub for QuickDraw
<code>intrfc/SANELib.text</code>	Standard Apple Numerics Environment (IEEE floating point).
<code>intrfc/SpeechIntf.text</code>	MacinTalk (speech synthesis)
<code>intrfc/ToolIntf.text</code>	ToolBox routines (Menu Mgr, Dialog Mgr, Window Mgr, ...)
<code>intrfc/WritelnWindow.text</code>	Debugging window (not for use in products)
<code>intrfc/WritelnWindow2.text</code>	Source to debugging window unit

## Object Files

These files are either for use by the Pascal compiler (indicated by \$USE), in which case they include the interface definition inside the object file, or for use by the linker (indicated by LINK), in which case they include the actual code to implement the interface, or for both.

obj/ABPasCalls.obj	AppleTalk implementation. LINK only.
obj/ABPasIntf.obj	AppleTalk definition. \$USE only.
obj/FixAsm.obj	Fixed point Math implementation (in assembler). Required for Graf3D. LINK with this.
obj/FixMath.obj	Fixed point Math definition. Required for Graf3D. \$USE only.
obj/Graf3D.obj	Definition for fixed point implementation of Graf3D (requires FixMath, does not require SANE). \$USE and LINK.
obj/Graf3DAsm.obj	Fixed point implementation of Graf3D (written in assembler). LINK with this.
obj/MacPrint.obj	MacPrint definition. \$USE only.
obj/MemTypes.obj	MemTypes definition. \$USE only.
obj/OSIntf.obj	OSIntf definition. \$USE only.
obj/OSTraps.obj	OSIntf implementation. LINK with this.
obj/PackIntf.obj	PackIntf definition. \$USE only.
obj/PackTraps.obj	PackIntf implementation. LINK with this.
obj/PasInit.obj	PasLib initialization implementation of %_BEGIN, %_END and %_TERM. LINK with this.
obj/PasLib.obj	PasLib implementation portion in Pascal. LINK with this.
obj/PasLibAsm.obj	PasLib implementation portion in assembler. LINK with this.
obj/PasLibIntf.obj	PasLib definition. \$USE only (if directly calling PasLib routines).
obj/PrLink.obj	MacPrint high-level implementation. LINK with this or obj/PrScreen, but not both; contains the following routines: PrClose, PrCloseDoc, PrClosePage, PrCtlCall, PrDrvrClose, PrDrvrOpen, PrError, PrintDefault, PrJobDialog, PrJobMerge, PrOpen, PrOpenDoc, PrOpenPage, PrPicFile, PrSetError, PrStlDialog, and PrValidate.
obj/PrScreen.obj	MacPrint low-level implementation. LINK with this or obj/PrLink, but not both; contains the following routines: PrCtlCall, PrDrvrClose, PrDrvrDCE, PrDrvrOpen, PrDrvrVers, PrError, PrNoPurge, PrPurge, and PrSetError.

obj/QuickDraw.obj	Quickdraw. \$USE and LINK.
obj/RTLib.obj	PasLib Run Time support--implementation of console I/O. LINK with this.
obj/SANELib.obj	SANE and Elems definition. \$USE only.
obj/SANELibAsm.obj	SANE and Elems implementation. LINK with this.
obj/SpeechAsm.obj	MacinTalk (speech synthesis) implementation (written in assembler). LINK with this.
obj/SpeechIntf.obj	MacinTalk (speech synthesis) definition. \$USE only.
obj/ToolIntf.obj	ToolIntf definition. \$USE only.
obj/ToolTraps.obj	ToolIntf implementation. LINK with this.
obj/WritelnWindow.obj	Debugging window (not for use in products). \$USE and LINK.

## Changes to Pascal Interfaces

The following changes were made to the Pascal interfaces since the February Software Supplement:

- 1) Version numbers have been added to all files. This release is version 1.1, corresponding to the final release of Inside Macintosh.
- 2) Pascal equate `swOverrunErr` has been corrected. The file system "PB" calls' async bit is now read correctly.
- 3) New equate ranges have been added to the Pascal interface: `evtQWhat` through `evtQMBut`, and `nsDrvErr` through `lastDskErr`.
- 4) The implementation of several Pascal routines were changed to make them smaller. These include `InitUtil`, `GetCaretTime`, `GetDoubleTime`, `FlushEvents`, `SetEventMask`, `MemError`, and `TEScrapHandle`. These routines are now declared as `INLINE $xxxx, $xxxx`; rather declared as external. Since the old interfaces required two words to call the routine in `ToolTraps` or `OSTraps`, all of the code required to implement the routine as well as the jump table entry is saved.
- 5) The implementation of several routines has been rewritten because they did not work correctly. If you reference any PB calls taking an async parameter, `SetSoundVol`, `GetVRefNum`, `SetAppBase` or `MoveHHi` it is recommended that you relink your program with the new `obj/OSTraps.obj`.
- 6) To use `Graf3D` from Pascal, `$USE obj/Graf3D.obj` but link with `obj/Graf3DAsm.obj`.
- 7) `RamSDOpen` and `RamSDClose` have been improved; they now arbitrate the serial ports correctly.
- 8) Calling `StartSound` asynchronously works better than it did previously.

The format of a few calls has been changed:

- 9) `MoveHHi` now returns its error via `MemError`.
- 10) The call `ScreenRes` has been added to return the screen's vertical and horizontal resolution in dots per inch.
- 11) Text Edit's filter routines `clikLoop` and `wordBreak` are now accessed from Pascal by `SetWordBreak` and `SetClikLoop`.

The purpose of `AsmClikLoop` was to allow a routine written in Pascal to be called while the mouse button was down. The Pascal routine had to be called `PasClikLoop` and the `clikLoop` field of the text edit record had to be set to `@AsmClikLoop`.

With the new `SetClikLoop` routine, the Pascal routine no longer has to be called `PasClikLoop`. So where before we would have the statement

```
hTE^^.clikLoop := @AsmClikLoop;  
we now have  
SetClikLoop(@MyClikLoop, hTE);
```

where the arbitrarily named routine `MyClikLoop` replaces `PasClikLoop`. Note that `AsmClikLoop` no longer exists.

### Note to developers using MacApp 0.2:

Recompile `UTReview2.text` after changing line 194 of that file from

```
anHTE^^.klikLoop := @AsmKlikLoop;  
to  
SetKlikLoop(@PasKlikLoop, anHTE);
```

## Assembler Equates

This supplement contains a new release of the equate and macro files needed for assembly language development for the Macintosh. This release is version 1.1 (the February Supplement contained a beta-release of version 1.1). Not too many changes have been made since the February release. The files are provided in both Lisa format (TLAsm files) and Macintosh (MDS, Macintosh 68000 Development System) format. The two sets of files are now completely consistent, the TLAsm files being mechanically produced from the MDS counterparts. These files should be the basis for all future Macintosh assembly language development.

The equates and macros are commented somewhat within the files themselves. More detailed documentation can be found in the appropriate sections of Inside Macintosh.

### Equate Files

<u>Lisa Workshop Files</u>	<u>MDS Files</u>	<u>Contents</u>
TLAsm/ATalkEqu.text	ATalkEqu.Txt	AppleTalk equates and globals
---	FixTraps.Txt	Fix-point math equates and globals (see <b>FixMath</b> and <b>Graf3D</b> section)
TLAsm/ FSEqu.text	FSEqu.Txt	File system equates and globals
---	Graf3D.Txt	Graf3D (3-D graphics) equates and globals (see <b>FixMath</b> and <b>Graf3D</b> section)
TLAsm/HardwareEqu.text	HardwareEqu.Txt	Hardware equates and globals (for debugging use only)
---	MacDefs.Txt	Macros translating Lisa Workshop assembler directives into MDS directives
---	MacTraps.Asm	Creates MacTraps.Sym (MDS symbol file)
TLAsm/PackMacs.text	PackMacs.Txt	Package macros
TLAsm/PrEqu.text	PrEqu.Txt	Printing equates and globals
TLAsm/QuickEqu.text	QuickEqu.Txt	QuickDraw equates and globals
TLAsm/QuickTraps.text	QuickTraps.Txt	QuickDraw traps
TLAsm/SANEMacs.text	SANEMacs.Txt	Numerics macros (see <b>SANE</b> section)
TLAsm/SysEqu.text	SysEqu.Txt	Low-level system equates and globals
TLAsm/SysErr.text	SysErr.Txt	System error numbers
TLAsm/SysTraps.text	SysTraps.Txt	Low-level system traps
TLAsm/ToolEqu.text	ToolEqu.Txt	Toolbox equates and globals
TLAsm/ToolTraps.text	ToolTraps.Txt	Toolbox traps

The files SysEqu, ToolEqu, and QuickEqu start with an equate such as "wholeSystem" which is used for conditional assembly. If you do not need the less common equates after ". IF wholeSystem" you can change wholeSystem to 0 and reduce the time and space required for your assembly. Note that two MDS symbol files are provided for each of these (e.g. SysEquX.D with wholeSystem on and SysEqu.D with wholeSystem off).



## Changes to Assembler Equate Files

The following changes to the assembly language equates have been made since the February Software Supplement:

- 1) Version numbers have been added to all files. This release is version 1.1, corresponding to the final release of Inside Macintosh.
- 2) The following equates correspond to a pre-release version of the Memory Manager, and were removed: FOnCheck, fChecking, mFulErr and memTrbBase. The FGZAlways and FBGZResrv flags were added, which allow additional control when the standard GrowZone function is called. GZCritical was obsoleted by the May 1984 system update, so it was removed.
- 3) New equates were added for the fonts: Times, Helvetica, Courier, Symbol, and Taliesin.
- 4) New equates have been added for sysPatListID, deskPatID, goodBye, and rdVerify.
- 5) Some equates have had minor cosmetic work done on them to make them consistent with the documentation. These include RestProc, renamed ResumeProc, and MrMacHook, renamed MBarHook. The equates commandMark, checkMark, diamondMark and appleMark were moved from menu equates to font equates. PrintVars was renamed PrintErr. SFSaveDisk and iPrSavPFil were added back into the public domain. The file HardwareEqu was beefed up, but please use this file for debugging only.
- 6) Assembly equates for AppleTalk have been added.
- 7) Some duplicate equates were removed. CurrPos, absPos, and relPos were removed because they duplicated fsAtMark, etc.
- 8) All instances of resource system references have been removed. These include: resSysRef, addRefFailed, rmvRefFailed, \_AddReference and \_RmveReference. These equates and traps are being removed because, to our knowledge, no one has found a use for them. **If you are using these features, contact Macintosh Technical Support immediately.** Otherwise, the corresponding code may be omitted from future systems.
- 9) The assembly equates dqEILnth, HasBundle, and invisible have been corrected.
- 10) To support arbitration of the serial ports, equates for ChooserBits, useExtClk, aPortUsed, and bPortUsed have been added.
- 11) The fixed point math, three dimensional graphics, speech synthesis, and floating point string conversion routines are now available for MDS users.

Application writers may find it useful to note that ApplScratch in TLAsm/ToolEqu is a 12 byte application scratch area in low memory.



## Workshop 3.9--New Features

### Workshop Shell (v. 3.9)

The shell has been enhanced to provide better support for hierarchical directories. Subdirectory support is now very solid. The **Copy**, **Backup**, and **Transfer** commands have been modified to create any needed subdirectories when a copy spills over onto more than one disk. The size of file names collected by the shell has been increased to prevent overflowing the file name strings when long file names with long subdirectory names are used. The **Equal** command in the **File Manager** now prints a summary of its comparisons. Exec files are now run with the three levels of prefixes rather than just the main prefix, which eliminates the need to put copies of common exec files in all your subdirectories. The **NEWER** command in the exec processor now works when files which do not exist are specified (allowing you to eliminate calls to the **EXISTS** function which verified that the arguments to **NEWER** existed, which should simplify your logic and speed up your exec files). Files which do not exist are assumed to have a date at the beginning of time (this does what you want). The **NEWER** function has also been optimized for speed. You can now call exec functions from the workshop **Run** command; the function result will be displayed on the console when the exec completes.

### Pascal Compiler (v. 3.76) and Code Generator (v. 3.65)

A few additional bug fixes have been added to the "Post-3.0" compiler described in the memo titled **Latest "Post-3.0" Lisa Pascal Compiler Enhancements** dated Feb. 8, 1985 (included with the February 1985 Software Supplement). Note that the SANE interface in Appendix A of that document is out of date (and has been omitted on new copies); see the file `intrfc/SANELib.text` on the **5/85 Workshop Supplement** disk for the latest interface.

If you are using an exec file other than the new `example/exec` file provided with this Supplement, please note the following: If you have Workshop 2.0 you should give the **\$M+** option to the code generator. If you have Workshop 3.0 or 3.9 you should give the **\$M+** option to the Pascal compiler and/or include it in your source code. If you are using Workshop 2.0 or an old version of the Workshop 3.0 Pascal compiler, make sure you also give the **\$X-** option to the compiler (this is optional when using **\$M+** in the new Workshop 3.9 compiler included in this Supplement).

### Linker (v. 0.9.3.1)

This linker supports all the functions necessary for development of Macintosh applications and drivers.

Using the Lisa Workshop linker, it is possible to "pre-link" a group of object files into a single object file for later linking convenience. Partial linking on the Lisa Workshop Linker is limited since the units being linked may not have any unit globals. To invoke this option, specify:

```
+R  modulename
```

as an option to the linker. This tells the linker that it will not find a main program in its set of input files and will cause it to generate a "raw" object file as output. If *modulename* is specified, then any procedures which are not called by *modulename* or anything that it calls will not be included in the link. Such code is considered to be "dead code" and will be stripped. If *modulename* is not specified, then no "dead code" will be stripped.

Once partial linking is done, all the code is collapsed into a single module with many entry points; since "dead code" stripping is done on a module basis, a library which has been partially linked will either be included *in its entirety* or not at all.

Note that when doing a partial link, the linker expects unresolved external references to be resolved in a later link; therefore no warnings for unresolved external references are given during partial links. Remember that if you are preparing an object file for RMaker there will be no later link.

Possible uses of partial linking include:

1. Writing code that will be input to RMaker as a raw object file (e.g. a desk accessory which must be a DRVR resource) in Pascal called by assembly language.
2. Building very large programs contained in too many files for the linker to handle.
3. Reducing link time by partially linking stable modules of an application and later linking only with changed modules.

If you have Workshop 3.0 or 3.9, please also note that your exec files should always give the +X option to the linker. This option is required for generating Macintosh code but it wasn't recognized by some linkers we've distributed in the distant past so you may have removed it from your exec files.

### **SANELib (v. 1.2)**

SANELib includes a regular unit called SANE that complements the extended-precision IEEE Standard arithmetic built into the new Pascal compiler described above. It also includes compile-time and run-time floating point support for that compiler. SANELib version 1.2 fixes a few minor bugs associated with version 0.9 (which was distributed with the February Software Supplement). For details see the release note titled **SANELib V1.2** included with this Supplement.

### **MacCom (v. 3.11)**

The `MacCom.obj` file on the **Workshop 3.9 Update Disk 1** is MacCom version 3.11. That version includes support for Macintosh/Lisa shared hard disks. This is provided through the command `AltDevice`. The `A` command can be used to tell Maccom to look on an alternate device (lower, paraport, upper) for the Macintosh directory. You can then move files to or from the specified Macintosh directory.

Maccom also supports conversion between Lisa and Macintosh text file formats. The command `Settings` displays a second command line:

`FinderInfo, RemoveSlashes, Tabs, ConvertText, MatchTypes, Status, Help, Quit`

`FinderInfo` and `RemoveSlashes` have the same effect as on the main command line (they were left in the main command line for exec file compatibility). `FinderInfo` here also allows you to change the defaults for the Finder type, creator, and bundle bit settings.

The `ConvertText` command allows for Lisa `.TEXT` file conversion. It asks you whether to convert to or from Lisa `.TEXT` files and what pathname extension (it need not be `.TEXT`) to use. This extension will be used to qualify filename searches when converting in either direction.

The `Tabs` command allows you to remove tabs (Macintosh to Lisa) or compress runs of blanks into tabs (Lisa to Macintosh) when processing text files with the `ConvertText` option.

The `MatchTypes` command allows you to qualify searches on Macintosh filenames by specifying a list of Finder types.

The `Status` command displays the current settings. `Help` displays help. `Quit` returns to the main command line.

## RMaker (v. 7.14b)

RMaker is the resource compiler for the Lisa Workshop. The format of RMaker input files (resource definition files) is described fully in the latest version of **Putting Together a Macintosh Application** (dated 5/5/85), which included with this Supplement.

This Supplement contains several versions of RMaker for users of Workshop 3.0 and 3.9. Running the Workshop 3.9 Update will install RMaker version 7.13b on your hard disk. The file 3.9only/RMaker7.14a.obj can be found on the **5/85 Workshop Supplement 2** disk; the file 3.9only/RMaker.obj on the **5/85 Workshop Supplement 3** disk is version 7.14b.

The latest versions are 7.14a and 7.14b. Both support larger resource files than earlier versions. RMaker version 7.14b will support even larger files than 7.14a, but at the expense of a smaller limit on the size of an item list (e.g. String Lists, DITLs, Pattern Lists, and Icon Lists). Version 7.14a will allow item lists which take up to 4K bytes of memory in RMaker; version 7.14b allows lists which take up to 2K bytes (enough for most applications). We recommend starting with version 7.14b and if the RMaker error "Fatal error. Data block overflow (item list too large)." is encountered, switching to version 7.14a (which should compile any resource definition files which worked with RMakers distributed with earlier Supplements).

Other enhancements to all of these versions of RMaker include:

Several bugs have been fixed.

Users may specify a meta-character as part of a menu item's text by repeating the meta-character twice; i.e., to put a left-parenthesis in a menu item, you should put (( in your resource definition file.

The processing of CODE resources has been optimized.

RMaker will capitalize any of Apple's "reserved resource type names" (e.g. ALRT, BNDL, and CODE) as listed in **Technical Note #32: Reserved Resource Types** (included with this Supplement) but will leave all other resource type names in their original case.

## Workshop Editor (Editor.obj dated 4/1/85)

The following enhancements have been made to the Workshop Editor:

**Prompts:** Prompts that require typed input now display default responses. You can get the default by typing <Return>. Typing <Clear> aborts the prompt.

**Markers:** The default response to **Set Markers** is now the first fourteen characters of the current selection. (Fourteen characters is the maximum length of a marker name). If the current selection matches the selection for any marker in the file, then that marker is the default response to **Delete Marker**.

**Find:** The default **Find...** target string is based on the current selection. If Search is Tokenized then the first token (delimited by spaces) is the default; otherwise, the default is the entire first line of the current selection.

**File Menu:** The default prompts for the File menu items (e.g. "Save a Copy in...", "Open...") are based on the current selection.

**Configurable Menus & Search Options:** When the editor first starts up it reads the file `--boot-edit.menus.text` to get the menu items. Users can edit this file but they should first back it up--major changes to this file could prevent the editor from functioning.

Changing this file (such as menu tiles and menu items) will take effect the next time the editor is started after it has been killed (either with the System Manager's **Manage Process** command or by exiting the Workshop).

Users who want to add new command-key equivalents to frequently used menus can do so by appending a menu item name with `"/x"` for some character `x` (e.g. "Throw Away Window/T" will make Apple-T equivalent to the Throw Away Window menu item).

Users who don't want the editor to start in "Search is Tokenized" mode can change the initial settings of the search options (Tokenized, Case Sensitive, and Wraparound). The editor now assumes that the search option menu items toggle between "Search is <Option>" and "Search is Not <Option>"; the presence of "Not" in the menu item name determines the startup configuration.

**Miscellaneous Editor enhancements:** Typing Apple-Period aborts printing. Markers are deleted with Cut (previously they were just hidden, now they are removed from the Marker list). The next event after cursor movement now works correctly.

## Assembler (v. 3.77):

The following enhancements have been made to to the Lisa Workshop assembler (version 3.77):

- 1) .SYM symbol files
- 2) .OUT output redirection
- 3) .CASEOUT case-sensitive linker symbols
- 4) .REF32 Directive
- 5) .REFA5 Directive

### 1) .SYM files

A mechanism similar to the Macintosh 68000 Development System assembler symbols file mechanism has been implemented in the Lisa Workshop assembler. This allows you to create a .SYM file containing the compressed definition of your symbols (i.e. Macintosh system equates) and then rapidly read these into your assembly.

To create the .SYM file: set up a separate assembly containing all definitions you want to include, then the dump statement. Format of dump statement is

```
.DUMP      filename
```

where filename is the name of the .SYM file (.SYM is automatically appended). Comments may occur on this line as well. A typical assembly is

```
.INCLUDE  TLASM/SYSEQU
.INCLUDE  TLASM/SYSERR
.INCLUDE  TLASM/SYSTRAPS
.INCLUDE  TLASM/TOOLEQU
.INCLUDE  TLASM/TOOLTRAPS
.DUMP     TLASM/EQUATES
.END
```

This creates the file TLASM/EQUATES.SYM, which is a compressed symbol file containing all the symbols defined in the listed files. Local labels are not included in the file. Forward references remain unresolved and should be avoided.

To use the .SYM file: Replace the corresponding definitions with an include of a .SYM file as follows:

```
.INCLUDE  TLASM/EQUATES.SYM
```

Note that the format is the same as the include of text files. For this reason, the .SYM suffix must be included in the name. (If the file is not found, the text file <name>.TEXT will be searched for.)

The effect of .INCLUDE ...SYM is to load the symbol table with the symbols previously defined. The upper/lower case characteristics are as defined at DUMP time. The INCLUDE statement does not cause checking for duplicate names or backpatching of values.

### 2) .OUT Output Redirection

The .OUT construct lets you change the assembler's output file between .PROC's or .FUNC's. This lets you create more than one output file from a single assembly; each file will contain a linker-legal object format with an integral number of assembly procs. The primary purpose of this feature is to combine assemblies that have common front ends, reducing assembly time.



Format:

```
.OUT  "filename"  
or  
.OUT  'filename'
```

where filename is the name of the new .OBJ file. The .OBJ suffix is automatically appended when not supplied. The .OUT command takes effect at the beginning of the next .PROC, .FUNC, or .MAIN, closing the existing output file and opening the new one (overwriting any previous file of the same name). Note that errors in opening the file may not get reported until the bottom of the next .PROC, due to the vagaries of the object file mechanism.

An '=' may be used as a wildcard in the file name. It stands for the original output file's volume and file name without extension. Therefore, if the first output file is -Volname - Temp.Obj, the command

```
.OUT  '=2'
```

will switch the output to the file -Volname-Temp2.Obj (a handy feature if you want to assemble to various volumes or subdirectories).

### 3) .CASEOUT case-sensitive linker symbols

The .CASEOUT directive causes ensuing newly defined symbols to be sent to the linker in their original case. It does NOT cause case-sensitivity during the assembly. This feature is intended to be used in linking assemblies to other case-sensitive languages, such as C.

Format:

```
.CASEOUT  
.NOCASEOUT
```

All symbols that are newly defined after .CASEOUT and up to a .NOCASEOUT will be sent in their original case to the linker. Typically, there is only a .CASEOUT at the beginning of the entire assembly. Note that CASEOUT applies to all symbols, including procs and funcs, defs and refs, and so on, although only some of the symbols are actually sent to the linker.

### 4 & 5) .REF32 and .REFA5 Directives

The assembler now offers symbolic access to UNIT global data from Workshop Pascal and EXTERNAL global data from Workshop C. It is now possible to reference data areas generated by high level languages and rely on the Linker to allocate and resolve the references. However, it is not possible to create new data areas or to specify initialization values. The assembler supports symbolic access with the directives .REF32 and .REFA5.

#### 4) .REF32 Directive

```
.REF32  NAMEX, NAMEY, NAMEZ
```

indicates to the assembler that NAMEX, NAMEY, and NAMEZ are the names of global data areas that will be based accessed via 32-bit (immediate) addresses. This should not be used with Workshop Pascal; it should only be of interest to users of Workshop C writing code to be executed on the Lisa. REF32 does not work in code executed on the Macintosh.

## 5) REFA5 Directive

.REFA5 NAMEA, NAMEB, NAMEC

indicates to the assembler that NAMEA, NAMEB, and NAMEC are the names of global data areas that will be based on register A5. For example, assume the following Pascal Unit interface (and implementation VARs):

```
unit NAMEA;

interface
var
  i: integer;
  L1: longint;
procedure PascalA;
procedure writeA;

implementation
var
  L2: longint;
```

One set of definitions to access variables i, L1, and L2 would be:

```
      .REFA5    NAMEA
i      .EQU     -2           ; 0 minus sizeof i
L1     .EQU     -6           ; start of i minus sizeof L1
L2     .EQU     -10          ; start of L1 minus sizeof L2
```

A more easily maintained set of definitions might be :

```
      .REFA5    NAMEA
FIRSTA .EQU     0
i      .EQU     FIRSTA-2     ; integer
L1     .EQU     i-4          ; longint
L2     .EQU     L1-4         ; longint
```

Notice that Unit variables are allocated in the negative direction from 0. Both INTERFACE and IMPLEMENTATION variables are accessible, although good programming practice would limit the references to only the INTERFACE variables.

Instructions to modify the variables i, L1, and L2 might be:

```
MOVE.W    #2, NAMEA+i    ; i := 2;
MOVEQ     #22, D0
MOVE.L    D0, NAMEA+L1   ; L1 := 22;
MOVE.W    #222, D0
EXT.L     D0
MOVE.L    D0, NAMEA+L2   ; L2 := 222
```

NOTES: when using symbols that have been declared REFA5, do not attempt to use the ...(A5) notation. (If you do, error numbers 21 and 72 will be reported by the assembler). The ...(A5) notation is still used for doing non-relocatable A5 references, e.g. when A5 is saved and some other value is loaded.

REFA5 with negative offsets works properly with Workshop Pascal (for code to be executed on the Macintosh or Lisa); it is not designed to work with Workshop C.

## .REFA5 Example Code Fragments

```
program SAMPLE;  
uses {$U NAMEA.OBJ} NAMEA;
```

```
procedure ASMA; External ;
```

```
begin  
  Pascal A;  
  WriteA;  
  AsmA;  
  WriteA;  
end.
```

```
      .PROC ASMA  
      .REFA5 NAMEA  
FIRSTA .EQU 0  
i       .EQU FIRSTA-2      ; integer  
L1      .EQU i-4           ; longint  
L2      .EQU L1-4          ; longint  
      MOVE.W #2,NAMEA+i    ; i := 2;  
      MOVEQ #22,D0  
      MOVE.L D0,NAMEA+L1    ; L1 := 22;  
      MOVE.W #222,D0  
      EXT.L D0  
      MOVE.L D0,NAMEA+L2    ; L2 := 222;  
      RTS  
      END
```

```
unit NAMEA;  
interface
```

```
  var i: integer;  
      L1: longint;
```

```
  procedure PascalA;  
  procedure writeA;
```

```
implementation
```

```
  var L2: longint;
```

```
  procedure PascalA;  
  begin  
    i := 1;  
    L1 := 11;  
    L2 := 111;  
  end;
```

```
  procedure writeA;  
  begin  
    writeln ( ' i = ',i:1, ' L1 = ',L1:1, ' L2 = ',L2:1);  
  end;
```

```
end.
```



# Resource File Builder (RFB)

## Introduction

RFB is a Resource File Builder tool for the Lisa Workshop. A version of RFB for Workshop 3.9 can be found in the file `3.9only/RFB.obj` on the **5/85 Workshop Supplement 3** disk. The source for the RFB tool (including a Workshop 3.9 exec file) can be found in `source/RFB.text` on the **5/85 Workshop Supplement 2** disk (see that file for more information). Workshop 2.0 users might be able to modify the source to make it work on their system.

In the course of development of "Please" (Hayes' database management package) for the Macintosh, it became apparent that a lot of time could be saved by avoiding RMaker resource compiles as much as possible. Further, as the program grew it was decided that it should be split into separate files, to allow the user to have only those resources on online disks that were necessary to perform the desired functions. None of the programs distributed with the Lisa Pascal Workshop proved sufficient to the task, so Toby Nixon, Software Analyst at Hayes, wrote the Resource File Builder (RFB) program.

RFB runs on the Lisa. It allows the developer to produce a Macintosh resource file on the Lisa from one or more other existing resource files. There are several specific advantages gained by using the program, including:

- Non-CODE resources, such as STR's, ALRTs, DLOGs, CTRLs, etc., can be placed in an RMaker input file and compiled once. A separate RMaker input file can be prepared which specifies ONLY the CODE resources. After each Link, only the CODE resource file must be reprocessed by RMaker. RFB combines the two resource files to produce the file that is MacCom'ed.
- Resources such as FONTs and PICTs, which are extremely difficult to produce in the Workshop, can be produced on the Mac (or Lisa under MacWorks), MacCom'ed onto the Workshop disk, and combined with other resources with RFB to produce the final Mac resource file.
- RFB allows an application to be easily split into separate resource files, such as a main file, utility file, and help file. The application running on the Mac can open the auxiliary files when necessary (usually because the user initiated a function that requires CODE or other resources from that file) using OpenResFile. The Mac Resource Manager automatically searches all open resources files, so no special code is needed in the application to find the resources once the files are open.
- When several programmers or other personnel (such as technical writers working on help text) are involved in the creation of non-CODE resources, they can separately process their work through RMaker, and distribute their files to the others. Each would run RFB to produce the executable version of the program, without having to wait for RMaker to process all of the other team members' work.

## Running RFB

After renaming `3.9only/RFB.obj` to `RFB.obj`, execute RFB by using the Workshop "R" (run) command. RFB is most often run from an exec file which includes all of the RFB input to produce the desired output. RFB requests the name of the output file to be produced. It does not check to see if the file already exists. It does not currently append any default extension, so the entire file name must be given. If a null entry is made, the program terminates.

RFB then requests the name of the input resource file to read. The entire file name must be entered. If the file is not found, an appropriate error message is displayed. If the file is found, it is opened and the resource map is read into memory. If no entry is made, the output resource file map is written, and the output file is closed. RFB then returns to allow the specification of another output file.

RFB then requests the resource type to be copied. If a null entry is made, RFB closes the input file and requests a new input file name. If "\*" is entered, then all resources of all types are copied from the input file to the output file. Otherwise, the type code entered is search for the the input file map, and an appropriate error message is displayed if it is not found. If it is found, an entry is made in the output file map for that type, and processing continues.

If the type specified is CODE, RFB asks if you want to specify code segments by name rather than number. This is very useful if the program is under active development, with frequent addition and deletion of code segments. If YES is specified, RFB then asks for the name of the linkmap file associated with the current input file. The linkmap file is opened and scanned, and the names and associated numbers of all code segments are tabulated.

After the type is entered, RFB requests the resource ID numbers to copy. If a null entry is made, RFB returns to request the next type (if no resources have been copied for the type, the type entry is removed from the output map). If "\*" is entered, all resources of the current type are copied to the output file. If CODE segments are being specified by name, the name is translated to the CODE resource ID number (segment number) for copying. If the specified resource is already in the output map, an error message is displayed.

While specifying CODE segments by name, it is still possible to specify by number. When a CODE segment name is requested, simply enter '#' followed by the number desired. This is necessary when requesting the jump table (#0) and the blank segment (#1). Resources of type VERS are handled a special way. When RFB starts, it gets the system date and time. When a VERS resource is copied, the resource data from the input file is replaced by the 10-byte system date-time stamp. All VERS resources written during a single run of RFB will have exactly the same time stamp data. This allows the application at run time to insure that auxiliary files are of the same version as the main program resource file, by simply doing a byte-by-byte comparison of the VERS resources in the files.

#### RFB Limits

RFB is currently limited by static array bounds to:

- 25 resource types in the output file.
- 128 references to any one type.
- 6144 bytes maximum input resource map size.
- 256 CODE segments in an input link map.

RFB files included on the **5/85 Workshop Supplement 2** and **3** disks:

- |                       |   |
|-----------------------|---|
| source/RFB.text       | Source code for RFB   |
| 3.9only/RFB.obj       | Object code for RFB   |
| 3.9only/RFB/exec.text | The procedure file used to build Please on the Macintosh, included as an example of an RFB exec file. |

Questions can be address to Toby Nixon of Hayes Microcomputer Products at (404) 449-8791.

## Example Programs and Exec Files

The files included on the **5/85 Examples 1** and **2** disks are example programs designed for use with the Lisa Workshop. These disks are in Macintosh format and the files are text-only files, so anyone with a Macintosh editor or word processor can read the source code (all but the largest can be read using the **File** editor example provided on the **5/85 MacStuff 2** disk). However the Lisa Pascal Workshop is required to compile the examples, most of which are written in Lisa Pascal.

If you plan to read the **Examples** disks *from a Macintosh*, note that they have no System folder so they cannot be used to boot a Macintosh; to use them you must boot another disk first. It may prove convenient to build disks with a copy of the System Folder from the **5/85 Mac Build Disk**, your editor, and files from the **Examples** disks that you want to read.

Macintosh 68000 Development System (MDS) users can assemble the DefProc (definition procedure) files. The **INCLUDE** statements must be modified, as described in the files. MDS may return minor warnings which may be ignored. Additional MDS examples can be found on the **5/85 MacStuff 4** disk.

Workshop exec files are provided to translate the Macintosh text only files to Lisa text files and copy them from Macintosh disks to a Lisa hard disk. Users who do not wish to copy all the examples (which require 1640 blocks of disk space) should move selected files onto another Macintosh disk or modify the exec files appropriately.

Workshop 3.0 or 3.9 users should copy the exec file `3.9only/example/convert/Mac2Lisa.text` from the **5/85 Workshop Supplement 3** disk to their hard disk, stripping off the "3.0only/" prefix. Invoke the file as follows:

**R<convert/Mac2Lisa**

The exec file will prompt the user for the two example disks and will invoke MacCom to copy and convert the files.

Workshop 2.0 users should copy the three `2.0only/` files from the **5/85 Workshop Supplement 2** disk to their hard disk, stripping off the "2.0only/" prefix. Then *for each example disk* invoke the exec file `Convert/Mac2Lisa1.text` as follows:

**R<convert/Mac2Lisa1**

This exec file takes several minutes to execute because after it copies the files using MacCom it must translate Macintosh text files to Lisa format (by invoking `Convert/Mac2Lisa2.text` which calls the program `Convert/TextConvert`). If you want to convert just a few files, see the exec files for more information; better yet, upgrade to Workshop 3.0.

Workshop 2.0 users should note that a few of the example programs will not compile and link on Workshop 2.0 without modification. These include: `3.9only/example/UNamAcc` (requires partial linking found in Workshop 3.9), `3.9only/example/Soundlab` (requires "Post-3.0 SANE" found in Workshop 3.9--Workshop 2.0 users should use `2.0only/example/Soundlab` from the February 1985 Software Supplement), `skel` (contains an optional `$USES SANE` statement which can be removed), `example/BoxSphere` (uses `BitSR` function found in the Workshop 3.9 compiler), and `Example/Event Tutor` (must be renamed without the blank to be used with Workshop 2.0 tools, and its source contains two uses of the Workshop 3.9 compiler features). The `fragment/AppleTalk...` files also must be renamed without a blank to be read from the Workshop 2.0 editor.

The example disks contain the following files:

Files on Mac disk 5/85 Examples 1:

Desk Accessory Example Sources

3.9only/example/UNamAcc.text  
3.9only/example/UNamAccAsm.text  
3.9only/example/UNamAccR.text  
3.9only/example/UNamAccX.text  
example/ADeskAcc.text  
example/ADeskAccR.text

Application Example Sources

3.9only/example/SoundLab.text  
3.9only/example/SoundLabR.text  
example/Boxes.text  
example/BoxesR.text  
example/BoxSphere.text  
example/BoxSphereR.text  
example/Control.text  
example/ControlR.text  
example/DebugWindow.text  
example/DebugWindowR.text  
example/Event Tutor.text  
example/Event TutorR.text  
example/File.text  
example/FileAsm.text  
example/FileR.text  
example/Grow.text  
example/GrowR.text  
example/Menu.text  
example/MenuR.text  
example/Modal.text  
example/ModalR.text  
example/Modal1.text  
example/Modal1R.text  
example/Modal2.text  
example/Modal2R.text  
example/Modeless.text  
example/ModelessR.text  
example/PicScrap.text  
example/PicScrapR.text

Source Fragments

fragment/AppleTalk 1.text  
fragment/AppleTalk 2.text  
fragment/AppleTalk 3.text  
fragment/AppleTalk 4.text  
fragment/ZoomRect.text  
ImageWriter/ResDef.text

Files on Mac disk 5/85 Examples 2:  
Application Example Sources

example/Print.text  
example/PrintR.text  
example/QDSample.text  
example/QDSampleR.text  
example/Samp.text  
example/SampR.text  
example/Scroll.text  
example/ScrollR.text  
example/Scroll.C.text  
example/SFSample.text  
example/SFSampleR.text  
example/ShowPaint.text  
example/ShowPaintR.text  
example/SineGrid.text  
example/SineGridR.text  
example/Skel.text  
example/SkelR.text  
example/SpeakFile.text  
example/SpeakFileR.text  
example/TextEdit.text  
example/TextEditR.text  
example/Window.text  
example/WindowR.text

DefProc Sources

defProcs/ButCDef.text  
defProcs/MDef.text  
defProcs/RDocWDef.text  
defProcs/SBarCDef.text  
defProcs/WDef.text



The files on these disks are example programs (how to use controls, windows, dialogs, etc.), example exec files, definition procedures, and sample desk accessories.

example/ADeskAcc.text  
example/ADeskAccR.text

Uriah, a sample desk accessory written in assembly language  
this resource definition file explains how to build it  
(there is no exec file)  
MDS users--see UriahMac.Asm on **5/85 MacStuff 4** instead

example/Boxes.text  
example/BoxesR.text

a Graf3D example

example/BoxSphere.text  
example/BoxSphereR.text

another Graf3D example

example/Control.text  
example/ControlR.text

a Control example

example/DebugWindow.text  
example/DebugWindowR.text

a WriteInWindow example

example/Event Tutor.text  
example/Event TutorR.text

a tool from Dartmouth to teach programmers about  
events; the resulting application it is worth looking at

example/File.text  
example/FileAsm.text  
example/FileR.text

the File example, a text processor, has lots of stuff in it  
an assembly module for File  
(note: File has been cleaned up since the Feb. Supplement)

example/Grow.text  
example/GrowR.text

a Window example

example/Menu.text  
example/MenuR.text

a Menu Manager example

example/Modal.text  
example/ModalR.text

a Modal Dialog example

example/Modal1.text  
example/Modal1R.text

another Modal Dialog example

example/Modal2.text  
example/Modal2R.text

and another Modal Dialog example

example/Modeless.text  
example/ModelessR.text

a Modeless Dialog example

example/PicScrap.text  
example/PicScrapR.text

an example of how to get pictures from the  
scrapbook

example/Print.text  
example/PrintR.text

a Print Manager Example

example/QDSample.text  
example/QDSampleR.text

a QuickDraw example

example/Samp.text  
example/SampR.text

the Samp example from Inside Macintosh

example/Scroll.text  
example/ScrollR.text

a Control Manager example--scrolling

example/SFSample.text  
example/SFSampleR.text

an example which displays a dialog that looks  
like Standard File (e.g. SFGGetFile)

example/ShowPaint.text  
example/ShowPaintR.text

how to unpack a MacPaint document

example/SineGrid.text  
example/SineGridR.text

another Graf3D example

example/Skel.text  
example/SkelR.text

a skeleton program from Dartmouth--  
definitely worth looking at

example/SoundLab.text  
example/SoundLabR.text

a Sound Driver example--uses SANE floating point in Pascal Workshop 3.9; may blow up on a Macintosh XL (better written sound software would not blow up)

example/SpeakFile.text  
example/SpeakFileR.text

a MacinTalk (speech synthesis) example

example/TextEdit.text  
example/TextEditR.text

a Text Edit example

example/Window.text  
example/WindowR.text

a Window Manager example

We have also included an example of how to write a desk accessory in Pascal. It's called UNamAcc. You need to have the new Workshop 3.9 partial linker to compile and link this.

3.9only/example/UNamAcc.text  
3.9only/example/UNamAccAsm.text  
3.9only/example/UNamAccR.text  
3.9only/example/UNamAccX.text

the Pascal source  
an assembly language portion  
RMaker source (resource definition file)  
the exec file for UNamAcc

There are several exec files for use with these examples on the **5/85 Workshop Supplement 3** disk. These exec files are designed to work with the 3.0 or 3.9 Workshops. If you are still running on the 2.0 Workshop then use the old 2.0only/ files. Note that not all of the above Pascal source files contain the \$M+ code generator option; that option must be included in the source and/or the exec file. We include it in our exec files.

The file 3.9only/example/Exec.text is a general purpose, semi-smart exec file that checks to see if the files you give it exist, and whether the files have changed since the last time you compiled or assembled. It sets the bundle bit in RMaker if (and only if) you give it a creator. To invoke it (after you rename 3.9only/= to =) you would type:

R<example/Exec(Pascal file, assembly file, resource file, creator, link file,  
source volume)

All of the items in the parentheses are optional. The default file is example/Window. The assembly file is assumed to be appended by 'Asm', and the resource file is assumed to be appended by 'R', like this:

Pascal Source	example/file.text
Assembly source	example/file <b>Asm</b> .text
RMaker source (resource definition)	example/file <b>R</b> .text

The link file is a file that contains the names of the files needed to link your program with. The default file name is example/MinLink.

3.9only/example/VanillaExec.text is an exec for those who would rather build without worrying about the details of whether the file has a creator or what files to link with. It assumes that the files are named using the conventions described above. It doesn't assemble. It links with everything. All of the files linked are commented as to when you would need them. It deletes the Desktop file when you run MacCom and it sets the bundle bit in RMaker whether you have a creator or not. To run it you enter:

R<example/VanillaExec(filename, creator)

There are also a set of files that allow you to build all of the examples at one time. They compile, link, make the resource file, and run MacCom to build each example and load it onto a Macintosh disk. They do not eject the disk until all the applications have been built. They do not work with the desk accessory examples.

3.9only/example/ExecAll.text  
3.9only/example/ExecAll2.text  
3.9only/example/ExampleList.text

this is the main exec--type R<example/ExecAll to run it  
This is very much like example/Exec, with some  
modifications  
This file has a list of each example filename, creator, and



link file if there is one. ExecAll reads this to determine what to exec.

Note that ExecAll can be invoked with any list of files in the format of ExampleList as follows:

R<example/ExecAll(MyExampleList.text)

There are also a number of LINK files. These contain filenames that are fed into the linker during the exec process.

3.9only/example/MinLink.text	The minimum of files needed to link.
3.9only/example/MaxLink.text	The maximum of files--everything.
3.9only/example/WritelnLink.text	The files needed for the WritelnWindow unit.
3.9only/example/Graf3DLink.text	The files needed for Graf3D.
3.9only/example/PrintLink.text	The files needed for printing.
3.9only/example/SANELink.text	The files needed for SANE.
3.9only/example/SpeechLink.text	The files needed for MacinTalk (speech synthesis).

Other files on the Examples disks include fragment/ files, defProcs/ files, example/Scroll.C.text and ImageWriter/ResDef.text. The fragment/ files are Lisa Pascal source fragments; these include the four Pascal AppleTalk example fragments from the final Inside Macintosh (better examples will appear in a future Technical Note) and a code fragment which draws the "zooming" rectangles which the Finder uses when opening and closing windows. The defProcs/ files are the assembly language definition procedures for the standard buttons, menus, scroll bars, windows, and round-cornered windows, included here in case you need to write your own custom definitions. The file example/Scroll.C.text is the Pascal program example/Scroll.text rewritten in a vanilla version of the language C (it may need modification before it can be compiled with a particular C compiler). The file ImageWriter/ResDef.text is the resource definition file fragment described in the enclosed document titled **The March 1985 ImageWriter: Programmers' Notes**.

## **MacWorks™ XL**

The **MacWorks XL 3.0** disk allows you start up any Macintosh XL or Lisa 2 system so it will run Macintosh software. The MacWorks XL disk included in the Supplement is version 3.0, the new version which Apple is releasing in July 1985. Any disks you received earlier may be pre-releases; make sure you test your software with version 3.0. Version 3.0 can be recognized because while it is being booted it displays the following:

MACWORKS XL 3.0  
COPYRIGHT 1985 - APPLE COMPUTER

MacWorks XL 3.0 allows direct startup from the hard disk and fixes numerous bugs that occurred with older versions of MacWorks. Version 2.0 of the Hard Disk Install tool (included in the Supplement) will allow you to format all or part of a built-in hard disk or one connected to the built-in parallel port. If you run Hard Disk Install and copy System and Finder to the hard disk, MacWorks will look for the hard disk automatically after starting up from the MacWorks disk. If you format your hard disk as a MacWorks-only disk (by choosing the "Don't Share" button) Hard Disk Install 2.0 will allow you to install or update MacWorks on your hard disk so that you can boot MacWorks without using any diskettes. To *force* MacWorks XL to start from a Macintosh 3 1/2" disk, hold down the Option key while you start your machine with MacWorks. Hard Disk Install and Parallel Printer Install can be found in the **MacXL Tools** folder on the **5/85 MacStuff 3** disk. For more information see the **MacWorks XL User Manual** and the enclosed documents **Technical Note #16 : MacWorks XL and Driver Bug in Pre-Release MacWorks XL**.

## Mac Build Disk

The Macintosh-formatted **5/85 Mac Build Disk** contains the System Folder that you should ship with your application. That folder contains the System file, the Finder (version 4.1), the ImageWriter driver, the Note Pad File, the Scrapbook File, and the Clipboard File. The first four of these are identical to the files on the 5/85 System Update Disk. The first three contain proprietary information and may not be distributed without specific written permission of Apple Computer, Inc. Licenses which permit distributing any or all of these are available for \$50 annually. Contact Apple's Software Licensing Department at (408) 973-4667 for more information.

This System file on this disk contains new printing software glue and a desk accessory called Choose Printer (version 1.3). These make it easy for users to choose which port (printer or modem) and printer (e.g. ImageWriter or LaserWriter) to use for printing. It will only display printers for which there is a print driver file on the startup disk; in addition, for printers connected over AppleTalk (e.g. LaserWriter) it will only display printers actually on the network.

This System file also contains new versions of Package 3 (Standard File), Package 4 (SANE Floating Point Numerics), and Package 5 (Transcendental Functions). Changes enhance performance and fix bugs (e.g. Package 5 no longer leaves itself locked) but will not affect applications.

The System version string (STR resource 0) has been changed to "Version 2.0 08-Apr-85".

The System file contains the following desk accessories: Alarm Clock, Calculator, Choose Printer, Control Panel, Key Caps, Note Pad, Puzzle, and Scrapbook. It also contains the following fonts: Chicago 12, Geneva 9, Geneva 12, and Monaco 9 (the minimum set of fonts required to run Macintosh software) and Monaco 12, New York 9, and New York 12. The above license entitles you to include other Apple fonts in your System file. Fonts and desk accessories can be added to or removed from the System file by use of the Font/Desk Accessory Mover (Font/DA Mover, which can be found on the **5/85 MacStuff 3** disk).

Note that the Mac Build Disk no longer includes an empty folder. Finder 4.1 has a *New Folder* menu item which allows users to create an empty folder when desired. You may still include an empty folder on your disk.

Also note that users could recreate the Note Pad File and Scrapbook File using the desk accessories provided. The Clipboard File can be created by applications or desk accessories. However, if no version of these files is distributed in the System Folder then the files will be created on the disk but not stored in any folder.

### Directory of Macintosh formatted disk **5/85 Mac Build Disk**

System Folder	System	80K	Mon, Apr 8, 1985
	Finder	47K	Mon, Apr 8, 1985
	Imagewriter	25K	Wed, Mar 6, 1985
	Note Pad File	2K	Sat, Apr 21, 1984
	Scrapbook File	1K	Tue, Apr 16, 1985
	Clipboard File	0K	Wed, Apr 10, 1985

## MacStuff Disks

The four Macintosh-formatted MacStuff disks contain various examples and tools that you can use on a Macintosh. Many of the files on the MacStuff disks have notes in their "Get Info" boxes which you can display by choosing *Get Info* from the Finder's File menu. Note that none of these disks contain the Finder (Finder 4.1 can be found on the **5/85 Mac Build Disk**). MacStuff disks 1, 2, and 3 have a MiniFinder (described in the **Macintosh Update for End-Users** document) and a System file which has been stripped of most fonts and desk accessories. Applications can be launched from the MiniFinder using the *Open* and *Open Other* buttons. The **5/85 MacStuff 4** disk cannot be used to boot a Macintosh; to use it you must boot another disk first. It may prove convenient to build disks with a copy of the System Folder from the **5/85 Mac Build Disk** and copies of the files from the **MacStuff** and **Examples** disks that you use frequently.

### Files on Mac disk 5/85 MacStuff 1:

System Folder	System MiniFinder Imagewriter
Tools	ResEdit REdit Localizer Dialog Creator EXAMPLE FreeTerm 1.6 Boot Configure Screen Maker DivJoin Printer

The **5/85 MacStuff 1** disk contains a number of tools and utilities of interest to Macintosh developers. Icons for all of these are displayed in the MiniFinder. Additional tools can be found on **MacStuff** disks 2 and 3.

ResEdit, REdit, and Dialog Creator are various flavors of resource editors. With these programs you can create and modify all kinds of resources including window templates, menus, dialogs, alerts, fonts, icons, bundles, and many more. ResEdit was written for programmers and provides many ways to modify resources. The Supplement contains ResEdit version 0.5 (this is a pre-release version, which means that you should use it with caution; back up a disk before attempting to modify it); a later version should be available from MAUG™ (via Compuserve) by the time you read this. REdit was written primarily for translators so not all modifications are allowed; however, it is very easy to use for certain modifications (e.g. changing the contents and size of a dialog box). Dialog Creator is useful when creating or editing RMaker resource definition functions for dialog and alert boxes. ResEdit is described in **ResEdit: A Macintosh Resource Editor**. REdit and Localizer are described in the booklet **Macintosh REdit: A Macintosh Resource Editor**. Dialog Creator and its example file EXAMPLE are described in **DIALOG CREATOR Instructions**.

FreeTerm 1.6 is a terminal emulator which, with a modem, will allow you to access services such as Compuserve (which hosts the users group MAUG™). It is described in the **FreeTerm** document. Note that FreeTerm 1.6 does not work correctly on pre-release versions of MacWorks XL; it will run on MacWorks XL 3.0 (which is included with this Supplement) but you may be forced to reboot to select a new serial port. FreeTerm 1.7 should work well on all version of MacWorks and should be available from MAUG™ (via Compuserve) by the time you read this.

BootConfigure allows you to display and set the contents of boot blocks on a 3 1/2" disk.

ScreenMaker converts the top left corner of a MacPaint document to a screen image (for use as a start up screen--a file named StartupScreen will be displayed when a disk is booted).

Printer is an old application which can be used to print files which have been spooled to disk or to set up the environment for printing in some other application.

DivJoin is a new application which allows you to divide a file that is larger than a diskette in size so that it can be moved from a hard disk onto multiple 3 1/2" diskettes. **IT ONLY DIVIDES THE DATA FORK OF THE FILE.** For example, if you have large text-only files on a hard disk, DivJoin can be used for moving them to other hard disks or backing them up onto diskettes.

To use DivJoin, open the file using the **Open** item in the File menu. This will bring up a window for that file. Select **Divide this File** from the Div/Join menu. DivJoin will then take your original file and divide it into sub-files, each the size of a diskette. If the original file is named `foo`, it will divide the original file into sub-files named `1.foo`, `2.foo`, `3.foo`, etc. It will create as many sub-files as needed. When DivJoin is finished dividing the file the window will disappear.

Take each sub-file and put it on a diskette. The last sub-file will probably be smaller than a full diskette. Later, when you want to recreate the original file, place all of the sub-files on the same hard disk volume. Start up DivJoin and **Open** `1.foo`. Choose **Join this File** from the Div/Join menu. DivJoin will create the file `foo` on the same volume. DivJoin does not support automatically feeding in diskettes for copying.

### Tools Which Have Been Removed

Some old tools which were included in previous Supplements have been omitted. They have been replaced by more reliable tools which are also more functional. The tools which were removed include: Resource Mover and IconEditor (replaced by ResEdit), Alert/Dialog Editor (replaced by REdit; Dialog Creator OR ResEdit could also be used), and Examine File, Set File, and Hex Dump (replaced by FEdit on the 5/85 MacStuff 2 disk).



## Files on Mac disk 5/85 MacStuff 2:

System Folder

System  
MiniFinder  
Imagewriter

Executable Examples

Boxes  
BoxSphere  
Control  
DebugWindow  
Event Tutor  
File  
Instructions  
More Info  
Grow  
Life  
Menu  
Modal  
Modal1  
Modal2  
Modeless  
PicScrap  
Print  
QDSample  
Samp  
Scroll  
SFSSample  
SineGrid  
ShowPaint  
MacPic  
Skel  
Bone  
SoundLab  
TextEdit  
Window

Disk Tools

Fedit  
Disk Utility

Switcher Folder

Switcher 3.0

MacDB & Nubs

MacDB  
MacNub A  
MacNub B  
WorksNub

The **5/85 MacStuff 2** disk has a folder of Executable Examples which are the result of building the example programs found on the **5/85 Examples 1 and 2** disks. The MiniFinder displays icons for a selection of these examples including Event Tutor, which is designed to help programmers to understand Macintosh events, and File, a simple text editor that can read multiple text files of up to 32K characters. Instructions and More Info are two File-readable text files which describe File; other data files for the examples which are found in the folder include MacPic (a MacPaint picture used by the ShowPaint tool), and Bone (an empty data file for Skel). The Executable Examples folder also contains the game Life (the source for which is not distributed).

The Disk Tools folder contains Disk Utility, an old tool that allows operations on 3 1/2" disks (it may not always work when running from a write protected disk), and Fedit, a file and disk edit utility program for the Macintosh patterned after the ZAP type programs available on many systems. Fedit is a powerful utility for use by average to highly technical users. It is not intended for the uninitiated user. The program allows the user low level, direct access to several types of disk volumes for both reading and updating. More information can be found in **Fedit: A File and Disk Editor**. Unlike most other programs on the Software Supplement, Fedit was not developed at Apple, but is shareware. If you find this program useful, we ask that you pay for it as requested by the author on the first screen of the program and on page 2 of its documentation.

The Switcher Folder contains Switcher 3.0, described in **Switcher (Beta Draft)** and **A Software Developer's Guide to Switcher**. Later versions of Switcher will be available from MAUG™ (via Compuserve) and, when released, through Apple dealers.

The MacDB & Nubs folder contains the latest version of the two-Macintosh debugger described in the chapter **The MacDB Debugger** distributed with a previous Software Supplement. For more information see that document or the **Debuggers** section of this document.



### Files on Mac disk 5/85 MacStuff 3:

System Folder

System  
MiniFinder

AppleTalk Tools

Peek  
Installer  
Poke  
Poke Packets

MacXL Tools

Hard Disk Install  
Parallel Printer Install

MacinTalk V1.1

MacinTalk  
ExceptionEdit  
SpeechLab  
SpeakFile  
TextToSpeak

Font Stuff

Font/DA Mover  
Fonts

The **5/85 MacStuff 3** disk has a System file with AppleTalk installed, to support the AppleTalk Poke tool. The AppleTalk Tools folder contains the tools Peek, Installer, and Poke and the Poke data file Poke Packets. For more information please refer to the enclosed documents **AppleTalk Information**, **AppleTalk Peek**, and **AppleTalk Poke**.

The MacXL Tools folder contains utilities that can be run under MacWorks XL. These utilities are described in the **MacWorks XL User Manual** (distributed with the February 1985 Supplement Update) and the **MacWorks XL** section of this document.

The MacinTalk V1.1 folder contains the MacinTalk driver file, the tools ExceptionEdit and SpeechLab, and the executable version of the example program SpeakFile with its data file TextToSpeak. MacinTalk is speech synthesis software for the Macintosh. For more information see the enclosed document titled **MacinTalk 1.1**.

The Font Stuff folder contains Font/DA Mover, the Font/Desk Accessory Mover tool, and Fonts, a corresponding data file; these can also be found on the May 1985 **System Disk**. The Font/DA Mover is described in the enclosed **Macintosh Update for End-Users** document.

## Files on Mac disk 5/85 MacStuff 4:

### MDS Stuff

#### .D Files

FSEqu.D  
Graf3D.D  
MacTraps.D  
QuickEqu.D  
QuickEquX.D  
SysEqu.D  
SysEquX.D  
ToolEqu.D  
ToolEquX.D

#### Trap Files

FixTraps.Txt  
MacTraps.Asm  
QuickTraps.Txt  
SysTraps.Txt  
ToolTraps.Txt

#### Equ Files

ATalkEqu.Txt  
FSEqu.Txt  
Graf3D.Txt  
HardwareEqu.Txt  
MacDefs.Txt  
PackMacs.Txt  
PrEqu.Txt  
QuickEqu.Txt  
SANEMacs.Txt  
SysEqu.Txt  
SysErr.Txt  
ToolEqu.Txt

#### .Rel Files

FixMath.Rel  
Graf3D.Rel  
PrLink.Rel  
SpeechAsm.Rel  
DEC2STR.Rel  
STR2DEC.Rel  
SDOpen.Rel

#### Graf3D Example

BoxSphere.Asm  
BoxSphere.Job  
BoxSphere  
BoxSphere.Link  
BoxSphere.Rel

#### UriahMac Desk Accessory Example

UriahMac.Asm  
UriahMac.R  
UriahMac.Link  
UriahMac

#### Resource Files

SERD  
ATalk/ABPackage

MacsBug Folder

Midibug  
Maxbug  
MacXLbug  
TermbugA  
TermbugB

Font file

Taliesin Font

Desk Accessory Examples

User Name Acc  
ADeskAcc (Uriah)

The **5/85 MacStuff 4** disk has no System Folder (so that it could include a large number of files). Therefore *it is not a bootable disk*; to use it you must boot another disk first.

That disk contains a number of files of interest to users of the Macintosh 68000 Development System (MDS); some are also of use to developers using other compilers with the MDS linker. These files for MDS are organized in six folders which are contained in the MDS stuff folder.

The .D Files, Trap Files, and Equ Files folders correspond to the folders of the same name on the **MDS2** disk (part of the MDS product). The folders on the **5/85 MacStuff 4** disk have the latest (version 1.1) assembly language equates (including ATalkEqu.Txt for AppleTalk), corresponding to the TLAsm/ files on the **5/85 Workshop Supplement 2** disk. See the **Assembler Equates** section of this document and the **Macintosh 68000 Development System User's Manual** for more information.

The .Rel Files folder contains assembled .Rel files which can be linked with the MDS linker. Many of these files provide access to functions previously available only from the Lisa Workshop.

The files FixMath.Rel and Graf3D.Rel support fixed point math and three-dimensional Quickdraw graphics; they are described in the **FixMath and Graf3D** section of this document.

The file PrLink.Rel contains the implementation of the high level printing routines; see the **Object Files** subsection of the **Workshop Pascal Interfaces** section of this document for more detailed information on PrLink.

The file SpeechAsm.Rel is the interface to the MacinTalk speech synthesis driver described in the document **MacinTalk 1.1**.

The files DEC2STR.Rel and STR2DEC.Rel are the numeric formatter and scanner (for conversion between decimal records and ASCII strings) described in the enclosed **SANE Numeric Scanner and Formatter** document.

The file SDOpen.Rel contains the routines RAMSDOpen and RAMSDClose described in the **RAM-Based Serial Drivers** section of this document.

The Graf3D Example folder contains BoxSphere, an example program written with MDS which uses the Graf3D (three-dimensional graphics) routines. See the **FixMath and Graf3D** section of this document for more information.

The UriahMac Desk Accessory Example folder contains source to Uriah, the example desk accessory, slightly modified to assemble with MDS on a Macintosh (the original Lisa assembler source can be found in the file example/ADeskAcc.text on the **5/85 Examples 1** disk). UriahMac, the file containing the resulting desk accessory, can be installed in a System file with the Font/DA Mover on the **5/85 MacStuff 3** disk.

The Resource Files folder contains two files containing resources which can be moved into an application's resource file using ResEdit. The resource file serd is described in the **RAM-Based Serial Drivers** section of this document. The resource file ATalk/ABPackage is described in the enclosed **AppleTalk Information** document.

The debuggers in the MacsBug Folder are documented briefly in their Get Info boxes and extensively in the **Debuggers** section of this document and the chapter **The MacsBug Debuggers** distributed with an earlier Supplement.

The Font File folder contains the new Taliesin Font, a pictorial font which appeared on the May 1985 System Update disk. It can be installed in a System file with the Font/DA Mover.

The Desk Accessory Examples folder contains the desk accessories produced from the two desk accessory examples whose source code appeared on the **5/85 Examples 1** disk. They can also be installed in a System file with the Font/DA Mover.

## FixMath and Graf3D

The Graf3D unit simulates three-dimensional graphics by making calls to QuickDraw. It can now be used from the Lisa Workshop, MDS, and other languages on the Macintosh which use the MDS linker. The version provided in this supplement uses fixed point arithmetic (which is smaller and faster than the floating point arithmetic used in some very old versions of Graf3D).

Programs written in Lisa Pascal using Graf3D must \$USE obj/FixMath and obj/Graf3D; they must link with obj/FixAsm and obj/Graf3DAsm (note that the files to link with are different than with the February Supplement). The file 3.9only/example/Graf3DLink.text on the **5/85 Workshop Supplement 3** disk contains the file names needed for this link.

Three Lisa Pascal example programs which use Graf3D are provided on the disks **5/85 Examples 1 and 2**: Boxes, BoxSphere, and SineGrid. Executable versions of these programs appear on the **5/85 MacStuff 2** disk.

Graf3D programs written on the Macintosh using MDS must *Include* FixTraps.Txt and Graf3D.Txt and *be linked with* FixMath.Rel and Graf3D.Rel. FixTraps.Txt is in the Trap Files folder, Graf3D.Txt is in the Equ Files folder, and FixMath.Rel and Graf3D.Rel are in the .Rel Files folder. BoxSphere.Asm, an assembly language version of the BoxSphere example, can be found in the Graf3D Example folder. That folder contains all the files needed to build the example using MDS as well as the resulting executable application. All of these folders are contained in the MDS Stuff folder on the **5/85 MacStuff 4** disk.

## SANE

SANE, the Standard Apple Numeric Environment, is embodied on the Macintosh in the Floating-Point Arithmetic Package (pack 4) and the Transcendental Functions Package (pack 5). These provide facilities for extended-precision floating-point arithmetic and advanced numerical applications programming. SANE is designed in strict accordance with IEEE Standard 754 for Binary Floating-Point Arithmetic.

Users of Lisa Pascal should not need to make calls to SANE directly; the **SANELib V1.2** document describes how the Workshop 3.9 Pascal compiler interacts with SANE.

To define the macros used by SANE, assembly language programmers must *Include* the file TLAsm/SANEMacs.text for the Lisa Workshop assembler or SANEMacs.Txt for the MDS assembler.

If you are using assembly language or a language without built-in support for SANE, you'll need to be familiar with the Apple Numerics Manual. This is the standard reference guide to SANE, and describes in detail how to call the Floating-Point Arithmetic and Transcendental Functions routines from assembly language. The Apple Numerics Manual is included in the Promotional ("phone book") Edition of Inside Macintosh and also in Apple // Assembly Language SANE (Apple product #A2W-0015).

Dec2Str and Str2Dec, SANE's numeric formatter and scanner (for conversion between decimal records and ASCII strings), are described in the enclosed **SANE Numeric Scanner and Formatter** document.

## RAM-Based Serial Drivers

For those who need the extra functionality of the RAM Serial Driver, the following two routines are supplied in `intrfc/OSIntf` (corresponding to `obj/OSIntf` and `obj/OSTraps`):

Function `RAMSDOpen(whichport: SPortSel):OSErr`;  
Procedure `RAMSDClose(whichport: SPortSel)`;

where `SPortSel=(SPortA,SPortB)`

These interfaces have not changed since the February 1985 Software Supplement, but the routines' implementations have been improved so that they now arbitrate serial ports (e.g. to prevent a conflict with AppleTalk). You should recompile any programs which use these calls.

The RAM Serial Driver can now be used from Lisa Pascal or assembler or the Macintosh 68000 Development System (MDS) assembler. When developing using the Lisa Workshop, copy the `serial/` files from the **5/85 MacSupplement 2** disk and move the text of `serial/AsyncR.text` into your resource definition file. When developing on a Macintosh using the MDS linker, link with `SDOpen.Rel` (on the **5/85 MacStuff 4** disk in the `.Rel Files` folder) and move the two SERD resources from the `SERD` file (on the **5/85 MacStuff 4** disk in the `Resource Files` folder) into your resource file (using the MDS RMaker or a resource editor).

Assembly language programmers should use the following code to bring in the RAM Serial Driver:

```
SPORTA EQU      $0000      ;".EQU" for Lisa assembler
SPORTB EQU      $0100

                XREF        RAMSDOpen    ;".REF" for Lisa assembler

                CLR.W        -(SP)        ; reserve space for function result
                MOVE.W       #SPORTB,-(SP) ; to select port B
                                           ; (use #SPORTA for port A)

                JSR          RAMSDOpen
                MOVE.W       (SP)+, D0     ; get the function result
```

Use the following code to close the driver:

```
                XREF        RAMSDClose   ;".REF" for Lisa assembler

                MOVE.W       #SPORTB,-(SP)
                JSR          RAMSDClose
```

`RAMSDOpen` loads and installs the Mac or MacXL RAM serial driver (resource type `SERD`, ID=1 for Mac, ID=2 for MacXL) if the system driver is version 0. The driver is then opened for both input and output. `RAMSDClose` must be called before the program ends to remove the RAM driver.

Possible errors from `RAMSDOpen` include:

-21...-23		- device manager error
-97	PortInUse	- some other driver is currently using this port
-98	PortNotCf	- parameter RAM is set for some other type use
-192	ResNotFound	- appropriate SERD resource not found
-108	MemFullErr	- not enough memory to load driver



# Debuggers

## MacsBug Debuggers

The **MacsBug Folder** contains five new versions of MacsBug (named Midibug, Maxbug, MacXLbug, TermbugA, and TermbugB). To use one of them, make a copy, rename it **MacsBug**, and reboot. Basic information about the MacsBug family of debuggers can be found in chapter 7 of the Macintosh 68000 Development System Users's Manual titled **The MacsBug Debuggers** which was distributed with an earlier Software Supplement. The information below describes changes to the debuggers since that manual was written.

- LisaBug is now called MacXLbug.
- xMacsbug is now called Midibug.
- MacXLbug (formerly LisaBug) is usable. No more bombing on mouse movement, etc. You can even interrupt during AT and HS commands .
- There are two new commands:

EA -- Exit to application...re-launch the current application.

DX -- Toggle debugger entry. Normally, if either the \$A9FF or \$ABFF A-trap is executed (two forms of the \$1FF debugger trap), program execution halts and the debugger is activated. DX allows you to control whether or not program execution halts. Note that the \$ABFF trap will still print a string; thus with debugger entry disabled, this causes an effect similar to the AT command (i.e. the Mac screen alternates between the debugger and the program).

- There are two other commands not mentioned in the MDS documentation.

SC -- Stack crawl. Assumes that LINK/UNLK A6 has been religiously performed at the beginning/end of each procedure/function (ala Pascal). The output format is as follows:

```
>SC
SF @<stack frame location> <address of call to procedure>
```

For example,

```
>SC
SF @0D633C ProcName+3A
```

means that the currently executing procedure/function has its local stack frame at \$D633C and was called from ProcName+\$3A (which is not the return address!).



**AR** -- A-trap record. The command has the same parameter format as **AB** (i.e. AR trap range, PC address range, D0 range). Whenever the parameter constraints are satisfied by an A-trap call, information about the call is recorded. The trap name, PC, A0, D0, and time are always saved. If the call was for an OS trap, 32 bytes pointed at by A0 are recorded, otherwise 32 bytes pointed at by A7 (stack ptr) are saved. To display the current saved information, type **AR** with no arguments. This command is especially useful for tracking down crashes in the Macintosh ROM. For example,

```
>AR 0 1000 @2AA @114
```

records traps 0 through 1000 (all traps) from ApplZone (\$2AA) through HeapEnd (\$114), so it will record the last trap call made from anywhere in the application heap (the application's code).

- A final feature is the ability to **IL** or **BR** on a procedure name. For example,

```
>IL ProcName+58
```

will disassemble code starting at 58 bytes (hex) into the procedure called 'ProcName', and

```
>BR ProcName+58
```

will set a breakpoint at the same location (this also works for **GT**, **ST**, **DM**, etc.). Note that the **\$D+** option must be specified in the Pascal source (the **PX** command in MacsBug can be used to disable/enable Pascal symbol display).

## **MacDB**

**MacDB** (the "two Mac debugger") can be used on one Macintosh to debug programs running on another Macintosh. **MacDB** and **MacNubs** can be found in the **MacDB & Nubs** folder of the **5/85 MacStuff 2** disk. This Supplement contains the versions of these files that were shipped with the Macintosh 68000 Development System product. One known bug is that **MacDB** indicates the target machine is a 128K Macintosh regardless of how much memory it actually has; this is relatively minor because **MacDB** can still reference all memory on the target machine. Information about **MacDB** can be found in chapter 6 of the Macintosh 68000 Development System Users's Manual titled **The MacDB Debugger** which was distributed with an earlier Software Supplement.

## Macintosh Product Availability List

The challenge of software distribution is faced by every developer: How to get information about your product to potential buyers and, more importantly, how to get your product through the distribution system and into the hands of the customer.

To help in this effort, the Apple Developers Group is constantly seeking better ways to "get the word out" on products that are available for the Macintosh. One very popular tool is our **Macintosh Product Availability List**. Published on a regular basis, it is distributed to over 7000 locations including our complete dealer base.

The folks at Menu.International, an International Software DataBase and fulfillment service, have been in the business of tracking software for some time. They currently have 20,000 products listed on their database and have shipping and purchasing arrangements for at least 3900 of these products.

Menu has created a special business unit to address the Macintosh market. We were so impressed with Menu's on-line service that we made arrangements for them to produce an enhanced version (including price and two-line product descriptions) of our Availability List to distribute to our dealers.

If you have a product that is shipping, call and confirm that Patti Barrus in the Developer Co-Marketing Group knows about you and your product. Patti's number is (408) 973-2972. Also, let the folks at Menu know so they can include you on the regular updates of the now famous Availability List. They can be reached at (303) 482-5000 or (800) Mac-Menu.

## Macintosh Pascal

Macintosh Pascal (the Pascal interpreter) has an undocumented copy protection scheme that causes an ExitToShell after 100 mouse clicks. If you've been running MacPascal off of a *copy* of your master disk, you've no doubt experienced this problem.

To rectify this problem, simply run off of one of your master disks (two were shipped in the package).

## MacApp

Apple is developing an object-oriented library called *MacApp™, The Expandable Application™*. MacApp implements the standard features common to most Macintosh programs. To write an application, you specify the differences between your particular product and MacApp. MacApp provides a substantial portion of a **Macintosh Application**, including standard objects like windows, menus, scroll bars, and documents, and standard operations like scrolling, resizing, printing, and text editing. It facilitates the implementation of Open, Save, and Undo, as well as multiple views of the same data. It makes it easy to conform with the user interface guidelines.

To benefit from MacApp a programmer should know Pascal and be just starting to develop a new Macintosh application. Using MacApp, such a programmer should be able to complete a Macintosh application in much less time than it would otherwise require.

At present, MacApp--and programs written to use it--must be written in Object Pascal, an object-oriented extension of Pascal. We currently have an "alpha" version of the Object Pascal compiler which runs on the Lisa Workshop 3.9 and generates code for the Macintosh. MacApp is also in an alpha-test state. To develop an application using the alpha Workshop version of MacApp you would need to have:

- (1) A Lisa (Macintosh XL) with at least 1 MB of memory and 5 MB of hard disk storage (10 MB recommended)
- (2) Lisa Pascal Workshop version 3.9 (Workshop 3.0 + 3.9 Update --> Workshop 3.9)
- (3) Inside Macintosh
- (4) The May 1985 Macintosh Software Supplement
- (5) A 512K Macintosh or a Macintosh XL with MacWorks XL 3.0  
( MacWorks XL 3.0 is included with the May Software Supplement)

If you have the hardware and software listed above and are interested in using MacApp, write to us at the address below; we will notify you when a released version of MacApp is available.

We would also like volunteers to test the alpha Workshop version of MacApp. Any Pascal programmer who will soon be starting to write a new application and would like to be an "alpha-tester" should submit a proposal of approximately three pages explaining the type of development planned and the reason (s)he would be a good tester. We will select a small number of alpha-testers.

Please send requests for notification and alpha-test proposals to:

MacApp Request  
c/o Eileen Crombie  
Apple Computer  
20525 Mariani Avenue  
Cupertino, CA 95014

Licenses to ship products built with MacApp will soon be available for a low annual fee.

## Smalltalk

We are currently developing a version of Smalltalk tailored to the Macintosh computer. In the meantime, in response to requests from several universities, we have released a "pre-product" version of the Smalltalk-80™ programming system running on the Macintosh and Macintosh XL computers. This is a fairly complete implementation of the Smalltalk-80 system, which we believe to be suitable for student projects and other preliminary experiments with Smalltalk.

Our current system is based on version 1 of the Xerox Smalltalk-80 system, and is language-compatible with the current (version 2) Xerox release. Our system generally follows the two Smalltalk-80 books by Goldberg and Robson. Several features will be found to be different or missing. We only provide enough documentation to get you started. If you are not already familiar with the Smalltalk-80 system, you will need further documentation not provided by Apple, such as the Goldberg and Robson books.

This pre-product release supplants an earlier March release that only ran on the Macintosh XL. Various options support stand-alone operation on the 512K Macintosh computer (minimal system), enhanced source-code access with hard disk or file server, and full system (32K objects) on machines with 1MB or more of memory, such as the Macintosh XL.

If you wish to try out Smalltalk, you may request an order form from the address below. We will include a more complete description of the products, pricing information, and an optional site license which permits the system to be installed and used throughout (but only within) your institution. Apple cannot provide support or documentation for this pre-product release. We are charging only enough to cover our costs of duplication and handling.

Smalltalk Request  
c/o Eileen Crombie  
Apple Computer, Inc.  
20525 Mariani Avenue  
Cupertino, CA 95014

CSNet: MacST@Apple.CSNET  
UUCP: {dual,nsc,voder,ios}!apple!MacST

Smalltalk-80 is a trademark of Xerox Corporation.

## Addresses

If you have technical comments regarding the Supplement, please write to us at:

Macintosh Technical Support  
Apple Computer  
Mail Stop 4-T  
20525 Mariani Avenue  
Cupertino, CA 95014

If you have questions about missing or damaged materials (disks or documentation), please contact our mailing facility at:

Apple Computer Mailing Facility/Milestone Group  
467 Saratoga Avenue, Suite 621  
San Jose, CA 95129

Customer Service:  
(408) 988-6009  
9:00 A.M. - 4:00 P.M., Pacific Time



 **ACCO.USA**  
WHEELING, ILLINOIS 60090

# 25102

MADE  
IN  
USA



0 50505 25102 5  
LT. BLUE/BLEU/AZUL CLARO